NodeBB Documentation

Release 1.x

The NodeBB Team (https://github.com/NodeBB/NodeBB/graphs/c

May 24, 2017

Contents

NodeBB is a next-generation discussion platform that utilizes web sockets for instant interactions and real-time notifications. NodeBB forums have many modern features out of the box such as social network integration and streaming discussions.

NodeBB is an open source project which can be forked on GitHub. If there are any discrepancies in the documentation please feel free to submit a pull request (via the "Edit on GitHub" button on the top right) or raise an issue on our issue tracker.

CHAPTER 1

Installing NodeBB

NodeBB Installation by OS

The following are step-by-step guides to help you get up and running.

Note: If your operating system is not listed here, please feel free to request a guide on our community or even better yet, submit one here.

Arch Linux

First, we install our base software stack. Be sure to *pacman -Syu* first to make sure you've synced with the repositories and all other packages are up to date.

```
$ sudo pacman -S git nodejs npm redis imagemagick icu
```

If you want to use MongoDB, LevelDB, or another database instead of Redis please look at the *Configuring Databases* section.

Next, clone this repository:

```
$ git clone -b v1.x.x https://github.com/NodeBB/NodeBB.git nodebb
```

To track the latest weekly build of NodeBB, substitute *weekly* in place of *v1.x.x*

Obtain all of the dependencies required by NodeBB:

```
$ cd nodebb
$ npm install
```

Initiate the setup script by running the app with the `setup` flag:

```
$ ./nodebb setup
```

The default settings are for a local server running on the default port, with a redis store on the same machine/port.

Lastly, we run the forum.

```
$ ./nodebb start
```

NodeBB can also be started with helper programs, such as `supervisor` and `forever`. *Take a look at the options here*.

Ubuntu

This installation guide is optimized for **Ubuntu 16.04 LTS** and will install NodeBB with MongoDB as the database. Fully patched LTS and equivalent **production** versions of software are assumed and used throughout.

Install Node.js

Naturally, NodeBB is driven by Node.js, and so it needs to be installed. Node.js is a rapidly evolving platform and so installation of an LTS version of Node.js is recommended to avoid unexpected breaking changes in the future as part of system updates. The Node.js LTS Plan details the LTS release schedule including projected end-of-life.

To start, add the nodesource repository per the Node.js Ubuntu instructions and install Node.js:

```
$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
$ sudo apt-get install -y nodejs
```

Verify installation of Node.js and npm:

```
$ node -v
$ npm -v
```

You should have version 6 of Node.js installed, and version 3 of npm installed:

```
$ node -v
v6.9.5
$ npm -v
3.10.10
```

Install MongoDB

MongoDB is the default database for NodeBB. As noted in the MongoDB Support Policy versions older than **3.x** are officially **End of Life** as of October 2016. This guide assumes installation of **3.2.x**. If Redis or another database instead of MongoDB the *Configuring Databases* section has more information.

Up to date detailed installation instructions can be found in the MongoDB manual. Although out of scope for this guide, some MongoDB production deployments leverage clustering, sharding and replication for high availibility and performance reasons. Please refer to the MongoDB Replication and Sharding topics for further reading. Keep in mind that NodeBB does not require any of these advanced configurations, and doing so may complicate your installation. Keeping it simple is best.

Abbreviated instructions for installing MongoDB:

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
$ echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.2 multiverse" |␣
↪sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list
$ sudo apt update && sudo apt install -y mongodb-org
```

Start the service and verify service status:

```
$ sudo service mongod start
$ sudo service mongod status
```

If everything has been installed correctly the service status should show as `active (running)`.

Configure MongoDB

General MongoDB administration is done through the MongoDB Shell `mongo`. A default installation of MongoDB listens on port `27017` and is accessible locally. Access the shell:

```
$ mongo
```

Switch to the built-in `admin` database:

```
{.sourceCode .} > use admin
```

Create an administrative user (**not** the `nodebb` user). Replace the placeholders `<Enter a username>` and `<Enter a secure password>` with your own selected username and password. Be sure that the < and > are also not left behind.

```
{.sourceCode .} > db.createUser( { user:  "<Enter a username>", pwd:  "<Enter
a secure password>", roles:  [ { role:  "readWriteAnyDatabase", db:  "admin"
}, { role:  "userAdminAnyDatabase", db:  "admin" } ] } )
```

This user is scoped to the `admin` database to manage MongoDB once authorization has been enabled.

To initially create a database that doesn't exist simply `use` it. Add a new database called `nodebb`:

```
{.sourceCode .} > use nodebb
```

The database will be created and context switched to `nodebb`. Next create the nodebb user and add the appropriate privileges:

```
{.sourceCode .} > db.createUser( { user:  "nodebb", pwd:  "<Enter a secure
password>", roles:  [ { role:  "readWrite", db:  "nodebb" }, { role:
"clusterMonitor", db:  "admin" } ] } )
```

The `readWrite` permission allows NodeBB to store and retrieve data from the `nodebb` database. The `clusterMonitor` permission provides NodeBB read-only access to query database server statistics which are then exposed in the NodeBB Administrative Control Panel (ACP).

Exit the Mongo Shell:

```
{.sourceCode .} > quit()
```

Enable database authorization in the MongoDB configuration file `/etc/mongod.conf` by uncommenting the line `security` and enabling authorization:

```
security:
  authorization: enabled
```

Restart MongoDB and verify the administrative user created earlier can connect:

```
$ sudo service mongod restart
$ mongo -u your_username -p your_password --authenticationDatabase=admin
```

If everything is configured correctly the Mongo Shell will connect. Exit the shell.

Install NodeBB

First, the remaining dependencies should be installed if not already present:

```
$ sudo apt-get install -y git build-essential
```

Next, clone NodeBB into an appropriate location. Here the home directory is used, though any destination is fine:

```
$ cd /opt
$ git clone -b v1.x.x https://github.com/NodeBB/NodeBB.git $HOME/nodebb
```

This clones the NodeBB repository from the `v1.x.x` branch to your home directory. A list of alternative branches are available in the NodeBB Branches GitHub page.

Obtain all of the dependencies required by NodeBB and initiate the setup script:

```
$ cd nodebb
$ npm install --production
$ ./nodebb setup
```

A series of questions will be prompt with defaults in parentheses. The default settings are for a local server listening on the default port `4567` with a MongoDB instance listening on port `27017`. When prompted be sure to configure the MongoDB username and password that was configured earlier for NodeBB. Once connectivity to the database is confirmed the setup will prompt that initial user setup is running. Since this is a fresh NodeBB install a forum administrator must be configured. Enter the desired administrator information. This will culminate in a `NodeBB Setup Completed.` message.

A configuration file *config.json* will be created in the root of the nodebb directory. This file can be modified should you need to make changes such as changing the database location or credentials used to access the database.

The last setup item is to configure NodeBB to start automatically. Modern linux systems have adopted systemd as the default init system. Configure nodebb to start via a systemd unit file at the location `/lib/systemd/system/nodebb.service`:

‘‘‘ {.sourceCode .} [Unit] Description=NodeBB forum Documentation=http://nodebb.readthedocs.io/en/latest/ After=system.slice multi-user.target

[Service] Type=simple User=nodebb

StandardOutput=syslog StandardError=syslog SyslogIdentifier=nodebb

Environment=NODE_ENV=production WorkingDirectory=/path/to/nodebb ExecStart=/usr/bin/node loader.js –no-daemon –no-silent Restart=always

[Install] WantedBy=multi-user.target

```
**Important**: Replace `/path/to/nodebb` with the correct path to your NodeBB␣
↪directory. If you followed this guide exactly, then you can `cd $HOME/nodebb &&␣
↪pwd` to see the absolute path to the directory, e.g.:
```

$ cd $HOME/nodebb && pwd /home/myusername/nodebb

$ ‘‘‘

Finally, enable and start NodeBB:

```
$ sudo systemctl enable nodebb
$ sudo service nodebb start
$ sudo service nodebb status
```

If everything has been installed and configured correctly the service status should show as `active`. Assuming this install was done on a Ubuntu Server edition without a desktop, launch a web browser from another host and navigate to the address that was configured during the NodeBB setup via IP address or domain name. The default forum should load and be ready for general usage and customization.

Debian

The current Ubuntu guide is not completely compatible with Debian and there are some specificities and especially the NodeJS installation, and how to get latest Redis.

Requirements

NodeBB requires the following software to be installed:

- Node.js at least 0.10 and greater
- Redis, version 2.6 or greater
- cURL installed, just do `sudo apt-get install curl` in order to install it

Node.js installation

Debian 7 and Debian 6 and older doesn't have *nodejs* packages included by default, but there are some solutions to install Node.js on your Debian distribution.

Wheezy Backport :

This solution is **ONLY for Debian 7**, simply run the following **as root** :

```
$ echo "deb http://ftp.us.debian.org/debian wheezy-backports main" > /etc/apt/sources.
↪list.d/wheezy-backports.list
$ apt-get update
```

To install Node.js + NPM, run this :

```
$ apt-get install nodejs-legacy
$ curl -L --insecure https://www.npmjs.org/install.sh | bash
```

The following install a Node.js version who is greater than 0.8 (at 29 March 2014 : 0.10.21)

Compiling from the source :

This solution is for Debian 6 (Squeeze) and greater, in order to install NodeJS, run this **as root** :

```
$ sudo apt-get install python g++ make checkinstall
$ src=$(mktemp -d) && cd $src
$ wget -N http://nodejs.org/dist/node-latest.tar.gz
$ tar xzvf node-latest.tar.gz && cd node-v*
$ ./configure
$ fakeroot checkinstall -y --install=no --pkgversion $(echo $(pwd) | sed -n -re's/.
↪+node-v(.+)$/\1/p') make -j$(($(nproc)+1)) install
$ sudo dpkg -i node_*
```

Get latest Software via DotDeb

Dotdeb is a repository containing packages to turn your Debian boxes into powerful, stable and up-to-date LAMP servers.

- Nginx,
- PHP 5.4 and 5.3 (useful PHP extensions : APC, imagick, Pinba, xcache, Xdebug, XHpro..)
- MySQL 5.5,
- Percona toolkit,
- Redis,
- Zabbix,
- Passenger...

Dotdeb supports :

- Debian 6.0 “Squeeze“ and 7 “Wheezy“
- both amd64 and i386 architectures

Debian 7 (Wheezy) :

For the complete DotDeb repositories :

```
$ sudo echo 'deb http://packages.dotdeb.org wheezy all' > /etc/apt/sources.list.d/
↪dotdeb.list
$ sudo echo 'deb-src http://packages.dotdeb.org wheezy all' >> /etc/apt/sources.list.
↪d/dotdeb.list
```

After this, add the following GPC keys :

```
$ wget http://www.dotdeb.org/dotdeb.gpg
$ sudo apt-key add dotdeb.gpg
```

And update your package source :

```
$ sudo apt-get update
```

Debian 6 (Squeeze)

For the complete DotDeb repositories :

```
$ sudo echo 'deb http://packages.dotdeb.org squeeze all' >> /etc/apt/sources.list
$ sudo echo 'deb-src http://packages.dotdeb.org squeeze all' >> /etc/apt/sources.list
```

After this, add the following GPC keys :

```
$ wget http://www.dotdeb.org/dotdeb.gpg
$ sudo apt-key add dotdeb.gpg
```

And update your package source :

```
$ sudo apt-get update
```

Installing NodeBB

Now, we have NodeJS installed and Redis ready to be installed, run this command for install the base software stack :

```
$ apt-get install redis-server imagemagick git build-essential
```

Next clone this repository :

```
$ cd /path/to/nodebb/install/location
$ git clone -b v1.x.x https://github.com/NodeBB/NodeBB.git nodebb
```

To track the latest weekly build of NodeBB, substitute *weekly* in place of *v1.x.x*.

Now we are going to install all dependencies for NodeBB via NPM :

```
$ cd /path/to/nodebb/install/location/nodebb (or if you are on your install location␣
↪directory run : cd nodebb)
$ npm install
```

Install NodeBB by running the app with *–setup* flag :

```
$ ./nodebb setup
```

1. ***URL of this installation*** **is either your public ip address or your domain name pointing to that ip address. Example:** `http://0.0.0.0` or `http://example.org`
2. **`Port number of your NodeBB`** **is the port needed to access your site: Note:** If you do not proxy your port with something like nginx then port 80 is recommended for production.
3. If you used the above steps to setup your redis-server then use the default redis settings.

And after all.. let's run the NodeBB forum

```
$ ./nodebb start
```

Note: If you NodeBB or your server crash, your NodeBB instance will not reboot (snap), this is why you should take a look at the other way to start your NodeBB instance with helper programs such as `supervisor` and `forever`, just *take a look here* it's simple as a click!

Extras, tips and Advice

You should secure your NodeBB installation, take a look here.

You should use Nginx (or similar) in order to reverse proxy your NodeBB installation on the port 80, *take a look here*

SmartOS

Requirements

NodeBB requires the following software to be installed:

- Node.js (version 0.10 or greater, instructions below).
- Redis (version 2.6 or greater, instructions below) or MongoDB (version 2.6 or greater).
- nginx (version 1.3.13 or greater, **only if** intending to use nginx to proxy requests to a NodeBB server).

Server Access

1. Sign in your Joyent account: Joyent.com
2. Select: `Create Instance`
3. Create the newest `smartos nodejs` image.

   Note: The following steps have been tested with images: `smartos nodejs 13.1.0` `smartos nodejs 13.2.3`
4. Wait for your instance to show `Running` then click on its name.
5. Find your `Login` and admin password. If the `Credentials` section is missing, refresh the webpage.

   Example: `ssh root@0.0.0.0` `A#Ca{c1@3`
6. SSH into your server as the admin not root: `ssh admin@0.0.0.0`

   Note: For Windows users that do not have ssh installed, here is an option: Cygwin.com

Installation

1. Install NodeBB's software dependencies:

   ```
   $ sudo pkgin update
   $ sudo pkgin install scmgit nodejs build-essential ImageMagick redis
   ```

   If any of those failed to install then:

   ```
   $ pkgin search *failed-name*
   $ sudo pkgin install *available-name*
   ```

2. **If needed** setup a redis-server with default settings as a service (automatically starts and restarts):

   If you want to use MongoDB, LevelDB, or another database instead of Redis please look at the *Configuring Databases* section.

   Note: These steps quickly setup a redis server but do not fine-tuned it for production.

   Note: If you manually ran `redis-server` then exit out of it now.

   ```
   $ svcadm enable redis
   $ svcs svc:/pkgsrc/redis:default
   ```

   Note: If the STATE is maintenance then:

   ```
   $ scvadm clear redis
   ```

- To shut down your redis-server and keep it from restarting:

```
$ scvadm disable redis
```

- To start up your redis-server and have it always running:

```
$ scvadm enable redis
```

3. Move to where you want to create the nodebb folder:

```
$ cd /parent/directory/of/nodebb/
```

4. Clone NodeBB's repository (you may replace the ending nodebb with a different folder name):

```
$ git clone -b v1.x.x https://github.com/NodeBB/NodeBB.git nodebb
```

- To track the latest weekly build of NodeBB, substitute *weekly* in place of *v1.x.x*.

5. Install NodeBB's npm dependencies:

```
$ cd nodebb
$ npm install
```

6. Run NodeBB's setup script:

```
$ ./nodebb setup
```

   (a) `URL used to access this NodeBB` is either your public ip address from your ssh `Login` or your domain name pointing to that ip address.

   Example: `http://0.0.0.0` or `http://example.org`

   (b) `Port number of your NodeBB` is the port needed to access your site:

   Note: If you do not proxy your port with something like nginx then port 80 is recommended for production.

   (c) `Please enter a NodeBB secret` - Do not email this or post publicly.

   (d) `IP or Hostname to bind to` - Use default unless your server requires otherwise.

   (e) If you used the above steps to setup your redis-server then use the default redis settings.

7. **Start NodeBB process manually: Note:** This should not be used for production but instead create a deamon manually, use Forever, or use Supervisor. *Take a look at the options here*.

```
$ node app
```

8. **Visit your app! Example:** With a port of 4567: `http://0.0.0.0:4567` or `http://example.org:4567`

   Note: With port 80 the `:80` does not need to be entered.

Note: If these instructions are unclear or if you run into trouble, please let us know by filing an issue.

Upgrading NodeBB

Note: Detailed upgrade instructions are listed in *Upgrading NodeBB*.

Windows 8

Required Software

First, install the following programs:

- https://windows.github.com/
- https://nodejs.org/en/download/
- http://imagemagick.org/script/binary-releases.php#windows/
- https://www.python.org/ftp/python/2.7.8/python-2.7.8.msi
- https://github.com/MSOpenTech/redis/releases
- https://www.microsoft.com/en-us/download/details.aspx?id=44914

You may have to restart your computer.

Running NodeBB

Start Redis Server and leave the command window that it starts in open

---

Note: The default location of Redis Server is

C:\Program Files\Redis\StartRedisServer.cmd

Open Git Shell, and type the following commands. Clone NodeBB repo:

```
git clone -b v1.x.x https://github.com/NodeBB/NodeBB.git
```

To track the latest weekly build of NodeBB, substitute *weekly* in place of *v1.x.x*

Enter directory:

```
cd NodeBB
```

Install dependencies:

```
npm install
```

Run interactive installation:

```
node app.js --setup
```

You may leave all of the options as default.

And you're done! After the installation, run

```
node app.js
```

and leave the window open.

You can visit your forum at `http://127.0.0.1:4567/`

Developing on Windows

It's a bit of a pain to shutdown and restart NodeBB everytime you make changes. First install supervisor:

```
npm install -g supervisor
```

Open up bash:

```
bash
```

And run NodeBB on "watch" mode:

```
./nodebb watch
```

It will launch NodeBB in development mode, and watch files that change and automatically restart your forum.

OSX Mavericks

Required Software

First, install the following programs:

- http://nodejs.org/
- http://brew.sh/

Running NodeBB

Install redis with homebrew:

```
brew install redis
```

Start redis server, in your terminal enter:

```
redis-server
```

Clone NodeBB repo:

```
git clone -b v1.x.x https://github.com/NodeBB/NodeBB.git
```

To track the latest weekly build of NodeBB, substitute *weekly* in place of *v1.x.x*.

Enter directory:

```
cd NodeBB
```

Install dependencies:

```
npm install
```

Run interactive installation:

```
./nodebb setup
```

You may leave all of the options as default, except "Which database to use (mongo)", which you should answer with "redis"

And you're done! After the installation, run

```
./nodebb start
```

You can visit your forum at `http://localhost:4567/`

CentOS 6/7

First we should make sure that CentOS is up to date, we can do so using this command:

```
yum -y update
```

**If your on CentOS 7, you will need to install the epel release, you can do so using the following command:

```
yum -y install epel-release
```

Now, we install our base software stack:

```
yum -y groupinstall "Development Tools"
yum -y install git redis ImageMagick npm
```

Now we need install nodejs via npm as the repo package is too old.

```
curl https://raw.githubusercontent.com/creationix/nvm/v0.13.1/install.sh | bash
source ~/.bash_profile
nvm list-remote
nvm install v0.12.7 # as of this writing check the result of the list-remote to see
↪all choices
```

Now start redis and set it to start on reboot

```
systemctl start redis
systemctl enable redis
```

If you want to use MongoDB, LevelDB, or another database instead of Redis please look at the *Configuring Databases* section.

Next, clone the NodeBB repository:

```
cd /path/to/nodebb/install/location
git clone -b v1.0.0 https://github.com/NodeBB/NodeBB nodebb
```

You'll want to replace `v1.0.0` with the (latest stable version), or `v1.x.x` if you'd like to set up the latest weekly build of NodeBB.

**Note: To clone the master branch you can use the same command with out the "-b" option.

After cloning the repository, obtain all of the dependencies required by NodeBB:

```
cd nodebb
npm install
```

Initiate the setup script by running the app with the `setup` flag:

```
./nodebb setup
```

The default settings are for a local server running on the default port, with a redis store on the same machine/port.

Assuming you kept the default port setting, you need to allow it through the firewall.

```
firewall-cmd --zone=public --add-port=4567/tcp --permanent
firewall-cmd --reload
```

Lastly, we run the forum.

```
./nodebb start
```

NodeBB can also be started with helper programs, such as `forever`. *Take a look at the options here*.

FreeBSD

This guide is for FreeBSD 10.1-RELEASE. It should work, with slight modifications, for all other relatively modern versions.

Make sure you are up to date, by running:

```
freebsd-update fetch
freebsd-update install
```

Install Redis:

```
pkg install redis
```

Follow the regular steps to run Redis at the runlevel, and start it.

Install gcc5:

```
pkg install gcc5
```

Install Node:

```
pkg install node
```

Clone NodeBB repo (this assumes you have *git* installed, otherwise use *pkg* to install it):

```
git clone -b v1.x.x https://github.com/NodeBB/NodeBB.git
```

To track the latest weekly build of NodeBB, substitute *weekly* in place of *v1.x.x*.

Enter directory:

```
cd nodebb
```

Install dependencies:

```
npm install
```

Run interactive installation:

```
./nodebb setup
```

You may leave all of the options as default.

You are finished! After the installation, run:

```
./nodebb start
```

Visit your forum at `http://FQDN:4567/` to finish configuration and setup. FQDN is the server fully qualified domain name.

- *Arch Linux*
- *Ubuntu*
- *Debian*
- *SmartOS*
- *Windows*
- *Mac*
- *CentOS*
- *FreeBSD*

Installing NodeBB on the Cloud

The following are step-by-step guides to help you get up and running on the cloud using popular PaaS solutions.

Note: If the PaaS solution of your choice is not listed here, please feel free to request a guide on our community or even better yet, submit one here.

Cloud 9 IDE

The following are installation instructions for the Cloud 9 web based IDE.

Step 1: Clone NodeBB into a new workspace from GitHub. You can use the following command from the terminal:

```
git clone -b v1.x.x https://github.com/NodeBB/NodeBB.git nodebb
```

To track the latest weekly build of NodeBB, substitute *weekly* in place of *v1.x.x*

The nodebb command after the git url will create a folder called nodebb so you have to `cd` into that directory after you have cloned NodeBB.

Step 2: Install redis with Cloud9's package manager

```
nada-nix install redis
```

Step 3: Run your redis server on port 16379 - port 6379 tends to be already used on Cloud 9. The "&" makes the command run in the background. You can always terminate the process later. $IP is a Cloud 9 system variable containing the global ip of your server instance.

```
redis-server --port 16379 --bind $IP &
```

Step 4: Find out your instance's ip address so NodeBB can bind to it correctly. This is one of Cloud 9's demands and seems to be the only way it will work. You can't use $IP in your config.json either (which means you can't enter $IP in the node app –setup).

```
echo $IP
```

Step 5: Install NodeBB and it's dependencies:

```
npm install
```

Step 6: Run the nodebb setup utility:

```
node app --setup
```

URL of this installation should be set to 'http://workspace_name-c9-username.c9.io', replacing workspace_name with your workspace name and username with your username. Note that as NodeBB is currently using unsecure http for loading jQuery you will find it much easier using http:// instead of https:// for your base url. Otherwise jQuery won't load and NodeBB will break.

Port number isn't so important - Cloud9 may force you to use port 80 anyway. Just set it to 80. If this is another port, like 4567, that is also fine.

Use a port number to access NodeBB? Again, this doesn't seem to make a big difference. Set this to no. Either will work.

Host IP or address of your Redis instance: localhost (the output of the $IP Command is also acceptable)

IP or Hostname to bind to: Enter what your $IP value holds here found in step 4. It should look something like: 123.4.567.8

Host port of your Redis instance: 16379

Redis Password: Unless you have set one manually, Redis will be configured without a password. Leave this blank and press enter

First-time set-up will also require an Admin name, email address and password to be set.

And you're good to go! Don't use the Run button at the top if the IDE, it has been a little buggy for me. Besides, you're better off using the command line anyway. Run:

```
node app
```

And then open http://workspace_name-c9-username.c9.io in your browser.

Troubleshooting

A common problem is that the database hasn't been started. Make sure you have set Redis up correctly and ran

```
redis-server --port 16379 --bind $IP
```

Heroku

Note: Installations to Heroku require a local machine with some flavour of unix, as NodeBB does not run on Windows.

1. Download and install Heroku Toolbelt for your operating system
2. Log into your Heroku account: `heroku login`

3. Verify your Heroku account by adding a credit card (at http://heroku.com/verify). *Required for enabling Redis To Go Add-on.*
4. Clone the repository: `git clone -b v1.x.x https://github.com/NodeBB/NodeBB.git /path/to/repo/clone`
    - To track the latest weekly build of NodeBB, substitute *weekly* in place of *v1.x.x*.
5. `cd /path/to/repo/clone`
6. Install dependencies locally `npm install --production`
7. Create the heroku app: `heroku create`
8. Enable Redis To Go for your heroku account (Nano is a free plan): `heroku addons:create redistogo:nano`
9. Run the NodeBB setup script: `node app --setup` (information for your Heroku server and Redis to Go instance can be found in your account page)
    - Your server name is found in your Heroku app's "settings" page, and looks something like `adjective-noun-wxyz.herokuapp.com`
    - Use any port number. It will be ignored.
    - Your redis server can be found as part of the redis url. For example, for the url: `redis://redistogo:h28h3wgh37fns7@fishyfish.redistogo.com:12345/`
    - The server is `fishyfish.redistogo.com`
    - The port is `12345`
    - The password is `h28h3wgh37fns7`
10. Create a Procfile for Heroku: `echo "web: node loader.js --no-daemon" > Procfile`
11. Commit the Procfile:

```
git add -f Procfile config.json package.json && git commit -am "adding Procfile and
↪configs for Heroku"
```

12. Push to heroku: `git push -u heroku v1.0.0:master` * Ensure that a proper SSH key was added to your account, otherwise the push will not succeed!
13. Initialise a single dyno: `heroku ps:scale web=1`
14. Visit your app!

If these instructions are unclear or if you run into trouble, please let us know by creating a topic on the Community Support Forum.

Keeping it up to date

If you wish to pull the latest changes from the git repository to your Heroku app:

1. Navigate to your repository at `/path/to/nodebb`
2. `git pull`
3. `npm install`
4. `node app --upgrade`
5. `git commit -am "upgrading to latest nodebb"`
6. `git push`

Nitrous.IO

The following are installation instructions for the *Nitrous.IO <http://nitrous.io>*.

Step 1: Create a new application in boxes with NodeJS :

https://www.nitrous.io/app#/boxes/new

Step 2: Open terminal / SSH to the application / Open IDE

Step 3: Get the files of NodeBB, unzip, delete master.zip and cd to the folder

```
wget https://github.com/NodeBB/NodeBB/archive/v1.x.x.zip && unzip NodeBB-v1.x.x.zip &&
↪ rm NodeBB-v1.x.x.zip && cd NodeBB-v1.x.x
```

Step 4: NPM Install

```
npm install
```

Step 5: Install Redis

```
parts install redis
```

Step 6: Setup NodeBB

```
./nodebb setup
```

Leave everything as default but you can change yourself.

I recommend the port number to bind : 8080

Step 14: And the last one, start NodeBB

```
./nodebb start
```

And then open the "Preview URI" without port if you have put for port : 8080.

Note

You can expand the resources of the application : http://www.nitrous.io/app#/n2o/bonus.

Openshift PaaS

Notice

A quickstart has been made to handle much of the below by using openshift's environment variables. If you're just getting started please take a look at ahwayakchih's openshift repo

The following are installation instructions for the Openshift PaaS. Before starting, you need to install Red Hat's rhc command line at https://developers.openshift.com/en/managing-client-tools.html

Step 1: Create an application:

You may do this through your web console or by using rhc:

```
rhc app create nodebb nodejs-0.10
```

If you're using the web console:

Scroll down and choose Node.js 0.10 in "Other types". Then click 'Next'. Type *nodebb* in application name or replace *(name)* to whatever you like.

You may see a note indicating you need to specify a namespace. Namespace can be anything as long you do not change the application name. If you do add a namespace make sure to use them in your RHC commands with -n and use -a to target you application by name.

```
rhc app create -a nodebb -n mynamespace nodejs-0.10
```

Scroll all the way down and click 'Create Application'. Then you need to wait for it to finish creating your first NodeBB application.

You will be asked if you want to code your application. Simply click 'Not now, contiune'. Then you will be redirected to an application page. It will tell you that it is created without any errors and it is started.

Step 2: Add a Database:

NodeBB works with eaither Redis or MongoDB, we'll use MongoDB.

```
rhc cartridge add mongodb-2.4 -a nodebb
```

In the web console:

Click 'see the list of cartridges you can add'. Choose the MongoDB cartridge. Then click 'Next'. You should see if you want to confirm. Click 'Add Cartridge'.

Step 3: Note your Database Credentials.

After installing the cartridge you'll get a notification of your username and password. Write it down somewhere, as you will need it later.

Open terminal (or Git Bash) and paste the following command to access SSH.

```
rhc app ssh -a nodebb
```

Note: If you got an error that it does not exist or similar, you need to do the following command and then try again.

```
rhc setup
```

Get your Database's Host, IP and Port

Save this for later as well...

```
echo $OPENSHIFT_NODEJS_IP && echo $OPENSHIFT_MONGODB_DB_HOST && echo $OPENSHIFT_
↪MONGODB_DB_PORT
```

In order: First line: NodeJS IP address - You should save it. Second line: Mongodb IP address - Write it down. Third line: Mongodb Port - Write it down.

Now exit SSH by pasting the following command.

```
exit
```

Note: You might have to type 'exit' once, and then again to exit SSH completely.

Step 4: Add NodeBB's Source Code on Openshift:

Go back to your web console and then click NodeBB application. Copy the URL address from "Scoure Code."

A similar scoure code URL address should be this: ssh://[code]@nodebb-[namespace].rhcloud.com/~/git/nodebb.git/

Go back to terminal. Paste the following command and then paste the URL address.

```
git clone ssh://[code]@nodebb-[namespace].rhcloud.com/~/git/nodebb.git/
```

Note: If it exists, check to make sure you do not have more than four files. If it is, delete it and rerun the command. Otherwise continue on.

Note: This will create NodeBB folder on your computer, usually ~/users/[name]/NodeBB

Then cd to NodeBB folder on your computer. And you will need to clone NodeBB from Github to your application by using this command. The default command git clone will not work with Openshift, unless you're in SSH. You may split up this command into two parts if you needed to clone your repository to another part of your computer start git bash from there.

```
cd nodebb && git remote add upstream -m master https://github.com/NodeBB/NodeBB.git
```

or

```
cd nodebb
git remote add upstream -m master https://github.com/NodeBB/NodeBB.git
```

Then pull files from NodeBB's repository.

```
git pull -s recursive -X theirs upstream v0.9.x
```

Openshift does not yet support version *1.0.0* or later, see this issue on github.

Step 5: Upload the source code to Openshift

Now you will need to commit and push files to your application's repository. Replace *message* with your message. It will take a while to finish.

```
git commit -a -m 'message' && git push
```

Step 6: Configure and Install NodeBB.

SSH back into your application:

```
rhc app ssh -a nodebb
```

Start the installation of NodeBB using interactive installer. You're going to fill in your application's details.

```
cd ~/app-root/repo && ./nodebb setup
```

Note: Web installer (npm start) will not work because the application has not been configured to bind to Openshift's allowed ports. We're about to do this right now.

URL used to access this NodeBB (http://localhost:4567) - Copy and paste your application's URL address and then add port 8080 like so: http://nodebb-{[}namespace{]}.rhcloud.com:8080

Please enter a NodeBB secret (code) - Just press enter.

Which database to use (redis) - enter *mongo*.

Host IP or address of your Mongo instance (127.0.0.1) - Copy & paste Mongo's IP address.

Host port of your Mongo instance (6379) - Copy & paste Mongo's port.

Password of your Mongo database - Enter your Mongo password.

Which database to use (0) - Enter the database name.

Step 7: Now you will need to edit config.json NodeBB had created. Paste the following command.

```
nano config.json
```

Add a line below "url" and then add the following. Repleace NodeJS IP Address to IP address of your application. Then exit the editor using CTRL+X.

` "bind_address": "NodeJS IP Address", `

Step 8: Now start your NodeBB on Openshift! And you're done! Then visit your website: http://nodebb-{[}namespace{]}.rhcloud.com/

```
cd ~/app-root/repo && ./nodebb start
```

Note

Starting, stopping, reloading, or restarting NodeBB now works on Openshift. Be sure you always do this before doing it. (Replace *[string]* with a nodeBB command of your choice). You can recover your application from critical plugin failures for example by running ./nodebb stop followed by ./nodebb reset -p nodebb-plugin-name

```
rhc app ssh -a nodebb
cd ~/app-root/repo
./nodebb [string]
```

If your application fails to start after a git push due to an error like EADDRINUSE openshift's application has critically failed and you may want to consider moving your NodeBB install to a new instance. Look up backing up and exporting databases. You can still resolve the error by first force stopping your application before making a push, but you will have to do this every single git push from now on.

```
rhc app-force-stop -a nodebb
git push origin master
```

Koding

Note: Installations to Koding requires a free account on Koding.com. (This guide has been changed to reflect the changes to Koding.com as of September 2014)

1. Create an account or log in to *Koding.com <http://koding.com>*
2. Click the Green Icon at the top that looks like >_
3. You will see your VM with Off to the right in red letters, click this, to power on your VM
4. Click it again when it says Ready.
5. You should now be inside a terminal window. The installation instructions are close to Ubuntu's, but vary slightly, as certain packages are already installed.
6. Firstly, we need to make sure we're up to date - `sudo apt-get update && sudo apt-get upgrade`
7. Enter your password you used to sign up, if you signed up using Github or another 3rd party, you will need to set one in your Account Settings. Then come back.
8. Now run the following `sudo apt-get install python-software-properties python g++ make`
9. Now we install NodeBBs other dependencies - `sudo apt-get install redis-server imagemagick`

10. Next, we clone NodeBB into a NodeBB folder - `git clone -b v1.x.x https://github.com/NodeBB/NodeBB.git nodebb` (Optional: Replace nodebb at the end if you want the folder to be a different name)
    - To track the latest weekly build of NodeBB, substitute *weekly* in place of *v1.x.x*.
11. Now enter the NodeBB folder - `cd nodebb` (unless you changed the foldername in the previous step, if you somehow forgot what you called it, run `ls` to see the name of the folder)
12. Now we install all the dependencies of NodeBB - `npm install` (could take a minute or two)
13. Set up nodebb using - `./nodebb setup`
14. The first setup question will ask for the domain name, this will vary, do not use localhost. Your domain name/Access URI is found on the left sidepanel by clicking the small icon to the right of your koding-vm-ID underneath VMS (it's a circle with 3 dots inside).
15. Complete the setup (defaults after the domain name are fine to accept, so press enter a few times until you get to "Create an Admin"
16. Create an Admin Username and password etc, it will then create categories and other things that make NodeBB awesome.
17. Now we can start NodeBB - `./nodebb start`
18. Open another tab in your browser of choice and navigate to `http://{uniqueID}.{yourkodingusername}.kd.io:4567` (assuming you didn't change the port number during setup)
19. You will see a screen to continue to your page, click the link about half way down to continue to your site.

Congratulations, you've successfully installed NodeBB on Koding.com

If these instructions are unclear or if you run into trouble, please let us know by filing an issue. (Be sure to mention @a5mith in your issue, as I wrote the guide)

Some issues with running on Koding

As Koding is free, it does come with some nuances to a regular cloud host:

1. Your VM will switch off after 60 minutes of inactivity. This doesn't mean the website unfortunately, but your Terminal Window (You can bypass this by keeping the terminal window open and running `ls` before your VM shuts down, alternatively, pay for the service and it will remain on)
2. It can be temperamental, sometimes you may receive "Your VM is unavailable, try again later", you can try logging out and back in, refreshing your page, or filing an issue with their support team.
3. Koding.com uses Ubuntu 14.04 to host your VM, so a basic knowledge of Ubuntu would always help.

Cloudron

The following are installation instruction for the Cloudron Cloudron is a platform for self-hosting web applications.

NodeBB is available on the Cloudron Store.

Your NodeBB installation is backed up and kept up-to-date automatically.

Code

The source code for the app is available here.

- Cloud9 (external)
- *Heroku*
- *Nitrous.IO*
- *Openshift*
- Digital Ocean (external)
- *Koding.com*
- Codio (external)
- Jelastic (external)
- RoseHosting.com - CentOS and Ubuntu (external)
- *Cloudron.io*

CHAPTER 2

Configuring NodeBB

The NodeBB Config (`config.json`)

The majority of NodeBB's configuration is handled by the Administrator Control Panel (ACP), although a handful of server-related options are defined in the configuration file (`config.json`) located at NodeBB's root folder.

Some of these values are saved via the setup script:

- `url` is the full web-accessible address that points to your NodeBB. If you don't have a domain, an IP address will work fine (e.g. `http://127.0.0.1:4567`). Subfolder installations also define their folder here as well (e.g. `http://127.0.0.1:4567/forum`)
- `secret` is a text string used to hash cookie sessions. If the secret is changed, all existing sessons will no longer validate and users will need to log in again.
- `database` defines the primary database used by NodeBB. (e.g. `redis` or `mongo`) – for more information, see *Configuring Databases*
- Both `redis` and `mongo` are hashes that contain database-related connection information, they contain some or all of the following:
  - `host`
  - `port`
  - `uri` (MongoDB only connection string)
  - `username` (MongoDB only)
  - `password`
  - `database`

The following values are optional, and override the defaults set by NodeBB:

- `port` (Default: `4567`) Specifies the port number that NodeBB will bind to. You can specify an array of ports and NodeBB will spawn port.length processes. If you use multiple ports you need to configure a load balancer to proxy requests to the different ports.

- `bcrypt_rounds` (Default: 12) Specifies the number of rounds to hash a password. Slower machines may elect to reduce the number of rounds to speed up the login process, but you'd more likely want to *increase* the number of rounds at some point if computer processing power gets so fast that the default # of rounds isn't high enough of a barrier to password cracking.
- `upload_path` (Default: `/public/uploads`) Specifies the path, relative to the NodeBB root install, that uploaded files will be saved in.
- **`jobsDisabled` This can be added to disable jobs that are run on a certain interval.** For example "jobsDisabled":true will disable daily digest emails and notification pruning. This option is useful for installations that run multiple NodeBB backends in order to scale. In such a setup, only one backend should handle jobs, and the other backends would set `jobsDisabled` to `true`.
- `socket.io` A hash with socket.io settings :
  - `transports` (Default: `["polling", "websocket"]`) Can be used to configure socket.io transports.
  - `address` (Default: `""`) Address of socket.io server can be empty
- **`bind_address` (Default: `0.0.0.0`, or all interfaces) Specifies the local address that NodeBB should bind to.** By default, NodeBB will listen to requests on all interfaces, but when set, NodeBB will only accept connections from that interface.
- `sessionKey` (Default: `express.sid`) Specifies the session key to use.

Configuring Databases

NodeBB has a Database Abstraction Layer (DBAL) that allows one to write drivers for their database of choice. Currently we have the following options:

MongoDB

If you're afraid of running out of memory by using Redis, or want your forum to be more easily scalable, you can install NodeBB with MongoDB. This tutorial assumes you know how to SSH into your server and have root access.

These instructions are for Ubuntu. Adjust them accordingly for your distro.

Note: If you have to add `sudo` to any command, do so. No one is going to hold it against you ;)

Step 1: Install MongoDB

The latest and greatest MongoDB is required (or at least greater than the package manager). The instructions to install it can be found on the MongoDB manual).

Step 2: Install node.js

Like MongoDB, the latest and greatest node.js is required (or at least greater than the package manager), so I'm leaving this to the official wiki. The instructions to install can be found on Joyent.

Note: NPM is installed along with node.js, so there is no need to install it separately

Step 3: Install the Base Software Stack

Enter the following into the terminal to install the base software required to run NodeBB:

```
# apt-get install git build-essential imagemagick
```

Step 4: Clone the Repository

Enter the following into the terminal, replacing */path/to/nodebb/install/location* to where you would like NodeBB to be installed.

```
$ cd /path/to/nodebb/install/location
$ git clone git://github.com/NodeBB/NodeBB.git nodebb
```

Step 5: Install The Required NodeBB Dependencies

Go into the newly created *nodebb* directory and install the required dependencies by entering the following.

```
$ cd nodebb
$ npm install
```

Step 6: Adding a New Database With Users

To go into the MongoDB command line, type:

```
$ mongo
```

To add a new database called *nodebb*, type:

```
> use nodebb
```

To add a user to access the *nodebb* database, type:

For MongoDB 2.6.x and 3.2.x

```
> db.createUser( { user: "nodebb", pwd: "<Enter in a secure password>", roles: [
↪"readWrite" ] } )
```

If you want to be able to view database statistics in NodeBB's admin control panel (Advanced → Database) type also this command:

```
> db.grantRolesToUser("nodebb",[{ role: "clusterMonitor", db: "admin" }]);
```

If you don't type the last command you will get this error message when trying to see database statistics:

```
Internal Error.

Oops! Looks like something went wrong!

/api/admin/advanced/database

not authorized on nodebb to execute command { serverStatus: 1 }
```

For earlier versions of MongoDB (if the above throws an error)

```
> db.addUser( { user: "nodebb", pwd: "<Enter in a secure password>", roles: [
↪"readWrite" ] } )
```

Note: NodeBB requires MongoDB 2.6.0 or higher. The role `readWrite` provides read or write any collection within a specific database to user.

Step 7: Configure MongoDB

Modify `/etc/mongodb.conf`.

```
# nano /etc/mongodb.conf
```

To enable authentication, type:

For MongoDB 2.6.x

Uncomment `auth = true`.

For MongoDB 3.2.x

Uncomment `security:` and add `authorization: enabled` below it (and don't forget to put two spaces before the second line). It should look like this:

```
security:
  authorization: enabled
```

Restart MongoDB.

```
# service mongodb restart
```

Step 8: Configuring NodeBB

Make sure you are in your NodeBB root folder. If not, just type:

```
$ cd /path/to/nodebb
```

To setup the app, type:

```
$ node app --setup
```

- Change the hostname to your domain name.
- Accept the defaults by pressing enter until it asks you what database you want to use. Type `mongo` in that field.
- Accept the default port, unless you changed it in the previous steps.
- Change your username to `nodebb`, unless you set it to another username.
- Enter in the password you made in step 5.
- Change the database to `nodebb`, unless you named it something else.

Continue with the installation, following the instructions the installer provides you.

Step 9: Starting the App

To start the app, run:

```
$ ./nodebb start
```

Now visit `yourdomainorip.com:4567` and your NodeBB installation should be running.

NodeBB can also be started with helper programs, such as *supervisor or forever*. You can also use `nginx` as a *reverse proxy*).

Advanced Settings

The mongodb nodejs driver has a default connection pool size of 5, if you need to increase this just add a *poolSize* setting into your config.json file under the mongo block.

- Redis (default, see *installation guides*)
- *Mongo*

Note: If you would like to write your own database driver for NodeBB, please visit our community forum and we can point you in the right direction.

Configuring Web Server / Proxies

Here a few options that you can use to proxy your NodeBB forum.

Configuring nginx as a proxy

NodeBB by default runs on port `4567`, meaning that builds are usually accessed using a port number in addition to their hostname (e.g. `http://example.org:4567`)

In order to allow NodeBB to be served without a port, nginx can be set up to proxy all requests to a particular hostname (or subdomain) to an upstream NodeBB build running on any port.

Requirements

- NGINX version v1.3.13 or greater
  - Package managers may not provide a new enough version. To get the latest version, compile it yourself, or if on Ubuntu, use the NGINX Stable or NGINX Development PPA builds, if you are on Debian, use DotDeb repository to get the latest version of Nginx.
  - To determine your nginx version, execute `nginx -V` in a shell

Configuration

NGINX-served sites are contained in a `server` block. This block of options goes in a specific place based on how nginx was installed and configured:

- `/path/to/nginx/sites-available/*` – files here must be aliased to `../sites-enabled`

- `/path/to/nginx/conf.d/*.conf` – filenames must end in `.conf`
- `/path/to/nginx/httpd.conf` – if all else fails

Below is the basic nginx configuration for a NodeBB build running on port `4567`:

```
server {
    listen 80;

    server_name forum.example.org;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;

        proxy_pass http://127.0.0.1:4567;
        proxy_redirect off;

        # Socket.IO Support
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

Below is another nginx configuration for a NodeBB that has port: `["4567","4568"]` in config.json.

```
server {
    listen 80;

    server_name forum.example.org;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;

        proxy_pass http://io_nodes;
        proxy_redirect off;

        # Socket.IO Support
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}

upstream io_nodes {
    ip_hash;
    server 127.0.0.1:4567;
    server 127.0.0.1:4568;
}
```

Below is an nginx configuration which uses SSL.

```
### redirects http requests to https
server {
    listen 80;
    server_name forum.example.org;

    return 302 https://$server_name$request_uri;
}

### the https server
server {
    # listen on ssl, deliver with speedy if possible
    listen 443 ssl spdy;

    server_name forum.example.org;

    # change these paths!
    ssl_certificate /path/to/cert/bundle.crt;
    ssl_certificate_key /path/to/cert/forum.example.org.key;

    # enables all versions of TLS, but not SSLv2 or 3 which are weak and now␣
↪deprecated.
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

    # disables all weak ciphers
    ssl_ciphers 'AES128+EECDH:AES128+EDH';

    ssl_prefer_server_ciphers on;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;

        proxy_pass http://127.0.0.1:4567;
        proxy_redirect off;

        # Socket.IO Support
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

Notes

- nginx must be on version 1.4.x to properly support websockets. Debian/Ubuntu uses 1.2, although it will work there will be a reduction in functionality.
- The `proxy_pass` IP should be `127.0.0.1` if your NodeBB is hosted on the same physical server as your nginx server. Update the port to match your NodeBB, if necessary.
- This config sets up your nginx server to listen to requests for `forum.example.org`. It doesn't magically route the internet to it, though, so you also have to update your DNS server to send requests for `forum.example.org` to the machine with nginx on it!

Configuring Nginx to use a custom error page

This example will demonstrate how to configure Nginx to use a custom 502 error page when your forum isn't running.

Create your custom error page

Create a new file `502.html` and place it in the `/usr/share/nginx/html` directory. This is where Nginx sets its document root by default. Be sure to add content to your `502.html` file. Here's an example which you can copy and paste:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Insert your page title here</title>
    </head>
    <body>
        <p>Insert your content here.</p>
    </body>
</html>
```

Configure Nginx to use your custom error page

We now need to tell Nginx to use our page when the relevant error occurs. Open your server block file `/etc/nginx/sites-available/default`. If you're using a non-default server block file, be sure to change `default`.

```
server {
    # Config will be here.

    error_page 502 /502.html;

    location = /502.html {
        root /usr/share/nginx/html;
        internal;
    }
}
```

The `error_page` directive is used so that the custom page you created is served when a 502 error occurs. The location block ensures that the root matches our file system location and that the file is accessible only through internal Nginx redirects.

Restart Nginx `sudo service nginx restart` and the next time a user visits your forum when it isn't running, they'll see your custom page.

Configuring apache as a proxy

Note: This requires Apache v2.4.x or greater. If your version of Apache is lower, please see :doc:'Apache v2.2.x Instructions <proxies/apache2.2>'

Enable the necessary modules

1. sudo a2enmod proxy
2. sudo a2enmod proxy_http

3. sudo a2enmod proxy_wstunnel
4. sudo a2enmod rewrite
5. sudo a2enmod headers

Add the config to Apache

The next step is adding the configuration to your `virtualhost.conf` file, typically located in `/etc/apache2/sites-available/`. The below configuration assumes you've used 4567 (default) port for NobeBB installation. It also assumes you have the bind address set to 127.0.0.1.

```
ProxyRequests off

<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

# edit the next line if you use https
RequestHeader set X-Forwarded-Proto "http"

RewriteEngine On

RewriteCond %{REQUEST_URI}  ^/socket.io            [NC]
RewriteCond %{QUERY_STRING} transport=websocket    [NC]
RewriteRule /(.*)           ws://127.0.0.1:4567/$1 [P,L]

ProxyPass / http://127.0.0.1:4567/
ProxyPassReverse / http://127.0.0.1:4567/
```

The last thing you need to be sure of is that the `config.json` in the NodeBB folder defines the node.js port outside of the url:

Example nodebb/config.json

```
{
    "url": "http://www.yoursite.com",
    "port": "4567",
    "secret": "55sb254c-62e3-4e23-9407-8655147562763",
    "database": "redis",
    "redis": {
        "host": "127.0.0.1",
        "port": "6379",
        "password": "",
        "database": "0"
    }
}
```

Change the domain and dont use the secret in the example above.

Configuring Apache v2.2.x as a reverse proxy to NodeBB

Prerequisites:

- `build-essential`

- `automake`
- `libtool`
- You can install these packages via apt

You need to manually compile and add the module `mod_proxy_wstunnel` to the Apache 2.2 branch. If you're running Ubuntu (prior to 14.04) or Debian, you're likely on the 2.2 branch of code.

Please use this guide to backport the `mod_proxy_wstunnel` module into the 2.2 code base of Apache:

http://www.amoss.me.uk/2013/06/apache-2-2-websocket-proxying-ubuntu-mod_proxy_wstunnel

Note: On ubuntu, if you're missing the ./configure file, you need to first run `./buildconf`. After this is complete, you will then be able to use `./configure`.

Configuring Varnish Cache

To be sure Varnish will work properly with NodeBB check that your configuration `/etc/varnish/default.vcl` is optimized for **websockets**.

```
backend nodebb {
  .host = "127.0.0.1"; # your nodebb host
  .port = "4567"; # your nodebb port
}

sub vcl_recv {

  # Pipe websocket connections directly to Node.js
  if (req.http.Upgrade ~ "(?i)websocket") {
    set req.backend = nodebb;
    return (pipe);
  }

  # NodeBB
  if (req.http.host == "forum.yourwebsite.com") { # change this to match your host
    if (req.url ~ "^/socket.io/") {
        set req.backend = nodebb;
        return (pipe); # return pass seems not working for websockets
    }
    return (pass); # don't cache
  }

}

sub vcl_pipe {
  # Need to copy the upgrade header
  if (req.http.upgrade) {
    set bereq.http.upgrade = req.http.upgrade;
  }
}
```

Configuring a node.js reverse proxy

In this tutorial we will create a reverse proxy https server complete with proxy rules, websockets, and TLS. This will allow multiple node applications to share the same domain, so that you can run NodeBB on a specific path (IE /forum) and another node application on another path.

Requirements

- NodeBB installation
- Node.js v5.0
- **The following npm packages installed using the command: npm install PACKAGE_NAME_HERE –save**
  - http-proxy-rules
  - express
  - http-proxy

1. Include packages

Create a node.js app with the following code

```
var https = require('https');
var httpProxy = require('http-proxy');
var express = require('express');
var HttpProxyRules = require('http-proxy-rules');
```

2. Define proxy rules and create proxy

Change these proxy rules to suit your needs. These rules will determine where traffic is proxied to based on the url path. In this example we assume you have an instance of NodeBB running on the default port

```
var proxyRules = new HttpProxyRules({
    rules: {
        '.*/docs': 'http://localhost:8081', // Rule (1) docs, about, etc
        '.*/docs/*': 'http://localhost:8081',
        '.*/about': 'http://localhost:8081',
        '.*/press': 'http://localhost:8081',
        '.*/jobs': 'http://localhost:8081',
        '.*/developers': 'http://localhost:8081',

        '.*/forum': 'http://localhost:4567/forum', // Rule (2) forums
        '.*/forum/*': 'http://localhost:4567/forum',
        '/forum/*': 'http://localhost:4567/forum',
        './forum/*': 'http://localhost:4567/forum',
        '/forum': 'http://localhost:4567/forum'
    },
    default: 'http://localhost:8081' // default target, will be landing page
});
var proxy = httpProxy.createProxy();
```

2. Change url in NodeBB config.json

Suffix the path you set in the proxy rules onto the default NodeBB url in the config.json file in your NodeBB directory. In this example, the path was /forum, so the URL becomes: .. code:: javascript

    http://localhost:4567/forum

3.Create the web server and call the proxy

First create the express.js app

```
var express = require('express');
var bodyParser = require('body-parser')
var mainapp = express();
mainapp.use(function(req,res,next){
    try{
        if (req.url.substr(0, 18).indexOf("socket.io")>-1){
            //console.log("SOCKET.IO", req.url)
            return proxy.web(req, res, { target: 'wss://localhost:4567', ws: true },␣
↪function(e) {
            //console.log('PROXY ERR',e)
            });
        } else {
            var target = proxyRules.match(req);
            if (target) {
                //console.log("TARGET", target, req.url)
                return proxy.web(req, res, {
                    target: target
                }, function(e) {
                //console.log('PROXY ERR',e)
                });
            } else {
                res.sendStatus(404);
            }
        }
    } catch(e){
        res.sendStatus(500);
    }
});
mainapp.use(bodyParser.json());
mainapp.use(bodyParser.urlencoded({ extended: false }));
```

Then put the code to start the web server, and put your HTTPS options in the options variable. (see node docs for more info about HTTPS)

Change the port (4433) to your port.

```
var options = {/*Put your TLS options here.*/};

var mainserver = https.createServer(options, mainapp);
mainserver.listen(4433);
mainserver.on('listening', onListening);
mainserver.on('error', function (error, req, res) {
    var json;
    console.log('proxy error', error);
    if (!res.headersSent) {
    res.writeHead(500, { 'content-type': 'application/json' });
    }

    json = { error: 'proxy_error', reason: error.message };
    res.end(JSON.stringify(json));
});
```

Thats it. Start up the proxy server, start up NodeBB, and start up your second server on port 8081 (or whichever port you chose)

- *Nginx*
- *Apache v2.4.x+*
- *Apache v2.2.x*
- *Varnish Cache*
- *Node.js*

CHAPTER 3

Running NodeBB

Running NodeBB

The preferred way to start and stop NodeBB is by invoking its executable:

- `./nodebb start` Starts the NodeBB server
- `./nodebb stop` Stops the NodeBB server
- Alternatively, you may use `npm start` and `npm stop` to do the same

The methods listed below are alternatives to starting NodeBB via the executable.

Upstart

Later version of Ubuntu may utilise Upstart to manage processes at boot. Upstart also handles restarts of scripts if/when they crash.

You can use Upstart to start/stop/restart your NodeBB.

Note: Prior to NodeBB v0.7.0, Upstart processes would not track the proper pid, meaning there was no way to stop the NodeBB process. NodeBB v0.7.0 includes some changes that allow Upstart to control NodeBB more effectively.

You can utilise this Upstart configuration as a template to manage your NodeBB:

```
start on startup
stop on runlevel [016]
respawn
setuid someuser
setgid someuser
script
        cd /path/to/nodebb
        node loader.js --no-silent --no-daemon
end script
```

From there, you can start stop and restart NodeBB as the root user: `start nodebb`, `stop nodebb`, `restart nodebb`, assuming `nodebb.conf` is the name of the Upstart config file.

Simple Node.js Process

To start NodeBB, run it with `node` (some distributions use the executable `nodejs`, please adjust accordingly):

```
$ cd /path/to/nodebb/install
$ node app
```

However, bear in mind that crashes will cause the NodeBB process to halt, bringing down your forum. Consider some of the more reliable options, below:

Supervisor Process

Using the supervisor package, you can have NodeBB restart itself if it crashes:

```
$ npm install -g supervisor
$ supervisor app
```

As `supervisor` by default continues to pipe output to `stdout`, it is best suited to development builds.

Forever Daemon

Another way to keep NodeBB up is to use the forever package via the command line interface, which can monitor NodeBB and re-launch it if necessary:

```
$ npm install -g forever
$ forever start app.js
```

Grunt Development

We can utilize grunt to launch NodeBB and re-compile assets when files are changed. Start up speed is increased because we don't compile assets that weren't modified.

Installing Grunt

```
$ npm install -g grunt-cli
```

Run grunt to start up NodeBB and watch for code changes.

```
$ grunt
```

CHAPTER 4

Upgrading NodeBB

Upgrading NodeBB

NodeBB's periodic releases are located in the Releases. These releases contain what is usually considered the most bug-free code, and is designed to be used on production-level instances of NodeBB.

You can utilise git to install a specific version of NodeBB, and upgrade periodically as new releases are made.

To obtain the latest fixes and features, you can also `git clone` the latest version directly from the repository (`master` branch), although its stability cannot be guaranteed. Core developers will attempt to ensure that every commit results in a working client, even if individual features may not be 100% complete.

As always, the NodeBB team is not responsible for any misadventures, loss of data, data corruption, or any other bad things that may arise due to a botched upgrade - so please **don't forget to back up** before beginning!

Upgrade Path

NodeBB's upgrade path is designed so that upgrading between versions is straightforward. NodeBB will provide upgrade compatibility (via the `--upgrade` flag) between the latest version of a lower branch and the latest version of the higher branch. For example, if `v0.2.2` is the latest version in the `v0.2.x` branch, you can switch to the `v0.3.x` branch and suffer no ill effects. Upgrading from `v0.2.0` to `v0.3.x` is not supported, and NodeBB will warn you when attempting to upgrade that you are not upgrading cleanly.

Upgrading between patch revisions

e.g. v0.1.0 to v0.1.1

Patch revisions contain bugfixes and other minor changes. Updating to the latest version of code for your specific version branch is all that is usually required.

Execute steps 1 through 3.

Upgrading between minor revisions

e.g. v0.1.3 to v0.2.0

Minor revisions contain new features or substantial changes that are still backwards compatible. They may also contain dependent packages that require upgrading, and other features may be deprecated (but would ideally still be supported).

Execute steps 1 through 4.

Upgrade Steps

Note: After upgrading between revisions (i.e. v0.0.4 to v0.0.5), it may be necessary to run the following upgrade steps to ensure that any data schema changes are properly upgraded as well:

1. Shut down your forum

While it is possible to upgrade NodeBB while it is running, it is definitely not recommended, particularly if it is an active forum:

```
$ cd /path/to/nodebb
$ ./nodebb stop
```

2. Back up your data

Note: This section is incomplete, please take care to back up your files properly!

Backing up Redis

As with all upgrades, the first step is to **back up your data**! Nobody likes database corruption/misplacement.

All of the textual data stored in NodeBB is found in a `.rdb` file. On typical installs of Redis, the main database is found at `/var/lib/redis/dump.rdb`.

Store this file somewhere safe.

Backing up MongoDB

To run a backup of your complete MongoDB you can simply run

    mongodump

which will create a directory structure that can be restored with the *mongorestore* command.

It is recommended that you first shut down your database. On Debian / Ubuntu it's likely to be: *sudo service mongodb stop*

Store this file somewhere safe.

Avatars

Uploaded images (avatars) are stored in /public/uploads. Feel free to back up this folder too:

```
cd /path/to/nodebb/public
tar -czf ~/nodebb_assets.tar.gz ./uploads
```

3. Grab the latest and greatest code

Navigate to your NodeBB: `$ cd /path/to/nodebb`.

If you are upgrading from a lower branch to a higher branch, switch branches as necessary. ***Make sure you are completely up-to-date on your current branch!***.

For example, if upgrading from `v0.3.2` to `v0.4.3`:

```
$ git fetch    # Grab the latest code from the NodeBB Repository
$ git checkout v0.4.x    # Type this as-is! Not v0.4.2 or v0.4.3, but "v0.4.x"!
$ git merge origin/v0.4.x
```

If not upgrading between branches (e.g. `v0.3.3` to `v0.3.4`, just run the following commands:

```
$ git fetch
$ git reset --hard origin/v0.3.x    # Replace v0.3.x with the branch name!
```

This should retrieve the latest (and greatest) version of NodeBB from the repository.

Don't know what branch you are on? Execute `git rev-parse --abbrev-ref HEAD` to find out.

Alternatively, download and extract the latest versioned copy of the code from the Releases Page. Overwrite any files as necessary. This method is not supported.

4. Run the NodeBB upgrade script

This script will install any missing dependencies, upgrade any plugins or themes (if an upgrade is available), and migrate the database if necessary.

```
$ ./nodebb upgrade
```

Note: `./nodebb upgrade` is only available after v0.3.0. If you are running an earlier version, run these instead:

- `npm install`
- `ls -d node_modules/nodebb* | xargs -n1 basename | xargs npm update`
- `node app --upgrade`

5. Start up NodeBB & Test!

You should now be running the latest version of NodeBB.

CHAPTER 5

Administrating NodeBB

Administrative Functions

The **Administrative Control Panel** (ACP) allows you to alter the behaviour of NodeBB, as well as customise various parts of its look and feel. Administrative functions such as user and group management are available from here.

The ACP is only accessible to administrators, and thus should be protected from unauthorised access whenever possible.

- Dashboard

Image Hosting APIs

Enabling Imgur Image Uploads

To enable post image attachments, install nodebb-plugin-imgur:

```
npm install nodebb-plugin-imgur
```

Follow the instructions on the plugin page: https://github.com/barisusakli/nodebb-plugin-imgur#setup

Uploading to Amazon S3

To enable automatic Amazon S3 file storage, install nodebb-plugin-s3-uploads-updated

```
npm install nodebb-plugin-s3-uploads-updated
```

Follow the instructions on the plugin page: https://github.com/LouiseMcMahon/nodebb-plugin-s3-uploads#nodebb-s3-uploads-plugin

CHAPTER 6

Scaling NodeBB

Scaling NodeBB

When it comes to maintaining many concurrent connections or high view rates, there are certain procedures that can be followed in order to streamline NodeBB.

This article attempts to outline the various strategies that can be employed by those finding that NodeBB is not running as fast as it should under full load.

Utilise clustering

By default, NodeBB will run on one process, and certain calls may take longer than others, resulting in a lag or queue for the same resources. To combat this, you can instruct NodeBB to run on multiple processes by adding the `port` property into your `config.json`:

```
{
    "port": ["4567", "4568", "4569"]  // will start three processes
}
```

Keep in mind you need to start nodebb with `node loader.js` or `./nodebb start` so that 3 workers can be spawned. Using `node app.js` will only use the first port in the array.

A proxy server like Nginx is required in order to load balance requests between all of the servers. Add an `upstream` block to your config:

```
upstream io_nodes {
    ip_hash;
    server 127.0.0.1:4567;
    server 127.0.0.1:4568;
    server 127.0.0.1:4569;
}
```

... and alter the `proxy_pass` value to read: `proxy_pass http://io_nodes;`

Use a proxy server to serve static assets

Nginx is exceedingly good at serving static resources (e.g. javascript, css, images, etc). By allowing Nginx to take over the task of serving this to the end user, the NodeBB process(es) will be left to handle only the API calls, which substantially lowers the amount of work for NodeBB (and increases your throughput).

Your Nginx config will need to be modified add the following `location` blocks:

```
location @nodebb {
    proxy_pass http://io_nodes;
}

location ~ ^/assets/(.*) {
    root /path/to/nodebb/;
    try_files /build/public/$1 /public/$1 @nodebb;
}

location /plugins/ {
    root /path/to/nodebb/build/public/;
    try_files $uri @nodebb;
}

location / {
    proxy_pass http://io_nodes;
}
```

Note: This configuration is only applicable to NodeBB versions v1.4.3 and above.

Furthermore, you can instruct Nginx to serve these assets compressed:

```
gzip            on;
gzip_min_length 1000;
gzip_proxied    off;
gzip_types      text/plain application/xml text/javascript application/javascript␣
↪application/x-javascript text/css application/json;
```

Sample Nginx configuration with all of the above applied

```
upstream io_nodes {
    ip_hash;
    server 127.0.0.1:4567;
    server 127.0.0.1:4568;
    server 127.0.0.1:4569;
}

server {
    listen 80;

    server_name community.nodebb.org;

    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_redirect off;

    # Socket.io Support
```

```
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";

    gzip            on;
    gzip_min_length 1000;
    gzip_proxied    off;
    gzip_types      text/plain application/xml text/javascript application/javascript␣
↪application/x-javascript text/css application/json;

    location @nodebb {
        proxy_pass http://io_nodes;
    }

    location ~ ^/assets/(.*) {
        root /path/to/nodebb/;
        try_files /build/public/$1 /public/$1 @nodebb;
    }

    location /plugins/ {
        root /path/to/nodebb/build/public/;
        try_files $uri @nodebb;
    }

    location / {
        proxy_pass http://io_nodes;
    }
}
```

Note: This configuration is only applicable to NodeBB versions v1.4.3 and above.

Configure Redis

When you setup NodeBB to use more than one process, it is advisable to configure Redis as well. Each NodeBB process can communicate with the others through Redis pub-sub. Install Redis on your server and add a `redis` block to your config.json. A sample `config.json` that uses mongodb as datastore and Redis for pubsub looks like this. When configured like this Redis will also be used as the session store.

```
{
    "url": "http://example.org",
    "secret": "your-secret-goes-here",
    "database": "mongo",
    "port": [4568,4569],
    "mongo": {
        "host": "127.0.0.1",
        "port": "27017",
        "database": "0"
    },
    "redis": {
        "host":"127.0.0.1",
        "port":"6379",
        "database": 0
    }
}
```

Configure Mongodb

A sample config.json to use mongodb replica sets

```
{
    "url": "http://forum.com",
    "secret": "secret",
    "database": "mongo",
    "mongo": {
        "host": "1.1.1.1,2.2.2.2,3.3.3.3",
        "port": "27017,27017,27017",
        "database": "myDbName?replicaSet=myReplSet",
        "options": {
            "server": {
                "socketOptions" : {
                    "keepAlive": 1000,
                    "autoReconnect": true
                }
            },
            "replSet": {
                "socketOptions": {
                    "keepAlive": 1000,
                    "autoReconnect": true
                }
            }
        }
    },
    "redis": {
        "host":"127.0.0.1",
        "port":"6379",
        "password":"",
        "database": 0
    }
}
```

CHAPTER 7

Contributing to NodeBB

NodeBB Style Guide

For the most part, NodeBB follows the Google Javascript Style Guide.

Code Formatting

Note: The existing codebase as of July 2013 does not adhere to this style guide 100%. If you see instances where the style guide is not adhered to, feel free to restyle and send off a pull request.

Indentation & Bracing

NodeBB uses tabbed indentation. Bracing should follow the One True Brace Style:

```
if (condition) {
    // code here ...
} else {
    // otherwise ...
}
```

Put conditionals and statements on separate lines and wrap with curly braces even if it's just one line:

```
if (leTired) {
    haveANap();
}
```

Errors

Most callbacks return an error as the first parameter. Handle this error first before processing further.

```
someFunction(parameters, function(err, data) {
    if (err) {
        return callback(err); // or handle error
    }
    // proceed as usual
});
```

Variables

Variables should always be prefaced with the *var* keyword:

```
var foo = 'bar';
```

Use var on multiple declarations :

```
var foo = 'bar';
var bar = 'baz';
```

Nomenclature

CamelCase if at all possible:

```
functionNamesLikeThis, variableNamesLikeThis, ClassNamesLikeThis, EnumNamesLikeThis,␣
↪methodNamesLikeThis, CONSTANT_VALUES_LIKE_THIS, foo.namespaceNamesLikeThis.bar, and␣
↪filenameslikethis.js.
```

Core Modules

Note: This section is under construction. Have a look at the modules folder for more information, located at:

```
public/src/modules
```

Alerts

The alert module is a toaster notification that can be called via the following syntax:

```
app.alert({
    title: 'Success!',
    message: 'Here\'s an example of an alert!',
    location: 'left-bottom',
    timeout: 2500,
    type: 'success',
    image: 'https://i.imgur.com/dJBzcGT.jpg'
});
```

The above code will result in this notification (default styling):

To style this, have a look at the vanilla theme's `modules/alert.less` and `templates/alert.tpl`.

Parameters:

1. `title` - string, which can be a language string as well. Some core language strings that you can use here include: `[[global:alert.success]]` and `[[global:alert.error]]`
2. `message` - string, which can be a language string as well.
3. `location` (optional) - `right-top` (default), `left-top`, `right-bottom`, `left-bottom`
4. `timeout` (optional) - integer in milliseconds, default is permanent until closed.
5. `type` - error, success, info, warning/notify
6. `image` (optional) - string, URL to image.
7. `closefn` (optional) - function. This is called when the user closes the alert via the (X) button.
8. `clickfn` (optional) - function. This is called when the user clicks on the alert.

CHAPTER 8

Plugin System

Writing Plugins for NodeBB

So you want to write a plugin for NodeBB, that's fantastic! There are a couple of things you need to know before starting that will help you out.

Like WordPress, NodeBB's plugins are built on top of a hook system in NodeBB. This system exposes parts of NodeBB to plugin creators in a controlled way, and allows them to alter content while it passes through, or execute certain behaviours when triggered.

See the full list of hooks for more information.

Filters and Actions

There are three types of hooks: **filters**, **actions**, and **static** hooks.

Filters act on content, and can be useful if you want to alter certain pieces of content as it passes through NodeBB. For example, a filter may be used to alter posts so that any occurrences of "apple" gets changed to "orange". Likewise, filters may be used to beautify content (i.e. code filters), or remove offensive words (profanity filters).

Actions are executed at certain points of NodeBB, and are useful if you'd like to *do* something after a certain trigger. For example, an action hook can be used to notify an admin if a certain user has posted. Other uses include analytics recording, or automatic welcome posts on new user registration.

Static hooks are executed and wait for you to do something before continuing. They're similar to action hooks, but whereas execution in NodeBB continues immediately after an action is fired, static hooks grant you a bit of time to run your own custom logic, before resuming execution.

When you are writing your plugin, make sure a hook exists where you'd like something to happen. If a hook isn't present, file an issue and we'll include it in the next version of NodeBB.

Configuration

Each plugin package contains a configuration file called `plugin.json`. Here is a sample:

```
{
    "url": "Absolute URL to your plugin or a Github repository",
    "library": "./my-plugin.js",
    "staticDirs": {
        "images": "public/images"
    },
    "less": [
        "assets/style.less"
    ],
    "hooks": [
        { "hook": "filter:post.save", "method": "filter" },
        { "hook": "action:post.save", "method": "emailme" }
    ],
    "scripts": [
        "public/src/client.js"
    ],
    "acpScripts": [
        "public/src/admin.js"
    ],
    "languages": "path/to/languages",
    "templates": "path/to/templates"
}
```

The `library` property is a relative path to the library in your package. It is automatically loaded by NodeBB (if the plugin is activated).

The `staticDirs` property is an object hash that maps out paths (relative to your plugin's root) to a directory that NodeBB will expose to the public at the route `/plugins/{YOUR-PLUGIN-ID}`.

- e.g. The `staticDirs` hash in the sample configuration maps `/path/to/your/plugin/public/images` to `/plugins/my-plugin/images`

The `less` property contains an array of paths (relative to your plugin's directory), that will be precompiled into the CSS served by NodeBB.

The `hooks` property is an array containing objects that tell NodeBB which hooks are used by your plugin, and what method in your library to invoke when that hook is called. Each object contains the following properties (those with a * are required):

- `hook`, the name of the NodeBB hook
- `method`, the method called in your plugin
- `priority`, the relative priority of the method when it is eventually called (default: 10)

The `scripts` property is an array containing files that will be compiled into the minified javascript payload served to users.

The `acpScripts` property is similar to `scripts`, except these files are compiled into the minified payload served in the Admin Control Panel (ACP)

The `languages` property is optional, which allows you to set up your own internationalization for your plugin (or theme). Set up a similar directory structure as core, for example: `language/en_GB/myplugin.json`.

The `templates` property is optional, and allows you to define a folder that contains template files to be loaded into NodeBB. Set up a similar directory structure as core, for example: `partials/topic/post.tpl`.

The `nbbpm` property is an object containing NodeBB package manager info.

Writing the plugin library

The core of your plugin is your library file, which gets automatically included by NodeBB if your plugin is activated.

Each method you write into your library takes a certain number of arguments, depending on how it is called:

- Filters send a single argument through to your method, while asynchronous methods can also accept a callback.
- Actions send a number of arguments (the exact number depends how the hook is implemented). These arguments are listed in the *list of hooks*.

Example library method

If we were to write method that listened for the `action:post.save` hook, we'd add the following line to the `hooks` portion of our `plugin.json` file:

```
{ "hook": "action:post.save", "method": "myMethod" }
```

Our library would be written like so:

```
var MyPlugin = {
        myMethod: function(postData) {
            // do something with postData here
        }
    };

module.exports = MyPlugin;
```

Using NodeBB libraries to enhance your plugin

Occasionally, you may need to use NodeBB's libraries. For example, to verify that a user exists, you would need to call the `exists` method in the `User` class. To allow your plugin to access these NodeBB classes, use `module.parent.require`:

```
var User = module.parent.require('./user');
User.exists('foobar', function(err, exists) {
    // ...
});
```

Installing the plugin

In almost all cases, your plugin should be published in npm, and your package's name should be prefixed "nodebb-plugin-". This will allow users to install plugins directly into their instances by running `npm install`.

When installed via npm, your plugin **must** be prefixed with "nodebb-plugin-", or else it will not be found by NodeBB.

Listing your plugin in the NodeBB Package Manager (nbbpm)

All NodeBB's grab a list of downloadable plugins from the NodeBB Package Manager, or nbbpm for short.

When you create your plugin and publish it to npm, it will be picked up by nbbpm, although it will not show up in installs until you specify a compatibility string in your plugin's `package.json`.

To add this data to `package.json`, create an object called `nbbpm`, with a property called `compatibility`. This property's value is a semver range of NodeBB versions that your plugin is compatible with.

You may not know which versions your plugin is compatible with, so it is best to stick with the version range that your NodeBB is using. For example, if you are developing a plugin against NodeBB v0.8.0, the simplest compatibility string would be:

```
{
    ...
    "nbbpm": {
        "compatibility": "^0.8.0"
    }
}
```

To allow your plugin to be installed in multiple versions of NodeBB, use this type of string:

```
{
    ...
    "nbbpm": {
        "compatibility": "^0.7.0 || ^0.8.0"
    }
}
```

Any valid semver string will work. You can confirm the validity of your semver string at this website: http://jubianchi.github.io/semver-check/

Linking the plugin

To test the plugin before going through the process of publishing, try linking the plugin into the node_module folder of your instance. https://docs.npmjs.com/cli/link

Using the terminal in the folder were you created your plugin, `/plugins/my-plugin`.

```
npm link
```

Then in the source folder were nodebb is installed.

```
npm link my-plugin
```

Your plugin should now be available in admin to be activated.

Adding Custom Hooks

You can use the same hooks sytem that NodeBB uses for plugins to create your own hooks that other plugins can hook into require the plugin librray in your code *var plugins = module.parent.require('./plugins');* and then use the *plugins.fireHook* command where ever you want them to be.

With this code any plugins can do things to the *postData* variable by hooking into the *filter:myplugin.mymethod* as they would a normall function. Once the plugins are done you can continue to work on the variable just as you normally would.

```
var Plugins = module.parent.require('./plugins');
var MyPlugin = {
        myMethod: function(postData) {
            // do something with postData here
            plugins.fireHook('filter:myplugin.mymethod', {postData : postData });
```

```
            // do more things with postData here
        }
    };
```

Testing

Run NodeBB in development mode:

```
./nodebb dev
```

This will expose the plugin debug logs, allowing you to see if your plugin is loaded, and its hooks registered. Activate your plugin from the administration panel, and test it out.

Disabling Plugins

You can disable plugins from the ACP, but if your forum is crashing due to a broken plugin you can reset all plugins by executing

```
./nodebb reset -p
```

Alternatively, you can disable a single plugin by running

```
./nodebb reset -p nodebb-plugin-im-broken
```

Available Hooks

The following is a list of all hooks present in NodeBB. This list is intended to guide developers who are looking to write plugins for NodeBB. For more information, please consult *Writing Plugins for NodeBB*.

There are two types of hooks, **filters**, and **actions**. Filters take an input (provided as a single argument), parse it in some way, and return the changed value. Actions take multiple inputs, and execute actions based on the inputs received. Actions do not return anything.

Important: This list is by no means exhaustive. Hooks are added on an as-needed basis (or if we can see a potential use case ahead of time), and all requests to add new hooks to NodeBB should be sent to us via the issue tracker.

As of 2014-10-08 we have moved the list of hooks into our wiki page. Please consult the list here.

Settings Framework

If you want to make your plugin customizable you may use the Settings Framework NodeBB offers.

Server-Side Access

First you need some default settings, just create a new object for this:

```
var defaultSettings = {
        booleans: {
            someBool: true,
            moreBools: [false, false, true]
        },
        strings: {
            someString: 'hello world',
            multiLineString: 'some\nlong\ntext',
            arrayOfStrings: ['some\nlong\ntexts', 'and another one']
        },
        numbers: {
            multiArrayDimensions: [[42,42],[21,21]],
            multiArrayDimensions2: [[42,42],[]],
            justSomeNumbers: [],
            oneNumber: 3,
            anotherNumber: 2
        },
        someKeys: ['C+S+#13'] // Ctrl+Shift+Enter
    };
```

Now you can use the server-side settings-module to access the saved settings like this:

```
var Settings = module.parent.require('./settings');
var mySettings = new Settings('myPlugin', '0.1', defaultSettings, function() {
    // the settings are ready and can accessed.
    console.log(mySettings === this); // true
    console.log(this.get('strings.someString') === mySettings.get().strings.
↪someString); // true
});
```

The second parameter should change at least every time the structure of default settings changes. Because of this it's recommended to use your plugins version.

To use the settings client-side you need to create a WebSocket that delivers the result of `mySettings.get()`.

The mySettings-object will cache the settings, so be sure to use methods like `mySettings.sync(callback)` when the settings got changed from somewhere else and `mySettings.persist(callback)` when you finished `mySettings.set(key, value)` calls.

You need to create a socket-listener like following to allow the admin to initiate a synchronization with the settings stored within database:

```
var SocketAdmin = module.parent.require('./socket.io/admin');
SocketAdmin.settings.syncMyPlugin = function() {
    mySettings.sync();
};
```

If you want to add a reset-functionality you need to create another socket-listener:

```
SocketAdmin.settings.getMyPluginDefaults = function (socket, data, callback) {
    callback(null, mySettings.createDefaultWrapper());
};
```

The methods of the `mySettings` object you probably want to use:

- `constructor()`
- **`sync([callback])`** Reloads the settings from database, overrides local changes.
- **`persist([callback])`** Saves the local changes within database.

- **get([key])** Returns the setting(s) identified by given key. If no key is provided the whole settings-object gets returned. If no such setting is saved the default value gets returned.
- **set([key,]value)** Sets the setting of given key to given value. Remember that it's just a local change, you need to call `persist` in order to save the changes.
- **reset([callback])** Persists the default settings.
- **getWrapper()** Returns the local object as it would get saved within database.
- **createWrapper(version, settings)** Creates an object like it would get saved within database containing given information and settings.
- **createDefaultWrapper()** Creates an object like it would get saved within database containing the default settings.

Client-Side Access

The next step is making the settings available to the admin.

You need to use the *hooks* `filter:admin.header.build` (to display a link to your page within ACP) and `action:app.load` (to create the needed route).

Within your page you can access the client-side Settings API via

```
require(['settings'], function (settings) {
    var wrapper = $('#my_form_id');
    // [1]
    settings.sync('myPlugin', wrapper);
    // [2]
});
```

To make a button with the id `save` actually save the settings you can add the following at `[2]`:

```
$('#save').click(function(event) {
    event.preventDefault();
    settings.persist('myPlugin', wrapper, function(){
        socket.emit('admin.settings.syncMyPlugin');
    });
});
```

As said before the server-side settings-object caches the settings, so we emit a WebSocket to notify the server to synchronize the settings after they got persisted.

To use a reset-button you can add the following at `[2]`:

```
$('#reset').click(function(event) {
    event.preventDefault();
    socket.emit('admin.settings.getMyPluginDefaults', null, function (err, data) {
        settings.set('myPlugin', data, wrapper, function(){
            socket.emit('admin.settings.syncMyPlugin');
        });
    });
});
```

There you go, the basic structure is done. Now you need to add the form-fields.

Each field needs an attribute `data-key` to reference its position within the settings. The Framework does support any fields whose jQuery-object provides the value via the `val()` method.

The plugin to use for a field gets determined by its `data-type`, `type` or tag-name in this order.

Additionally the following plugins are registered by default:

- **array (types: div, array)** An Array of any other fields. Uses the object within `data-attributes` to define the array-elements. Uses `data-new` to define the value of new created elements.
- **key (types: key)** A field to input keyboard-combinations.
- **checkbox, number, select, textarea** Handle appropriate fields.

A full list of all attributes that may influence the behavior of the default Framework:

- data-key: the key to save/load the value within configuration-object
- data-type: highest priority type-definition to determine what kind of element it is or which plugin to associate
- type: normal priority type-definition
- data-empty: if `false` or `0` then values that are assumed as empty turn into null. data-empty of arrays affect their child-elements
- data-trim: if not `false` or `0` then values will get trimmed as defined by the elements type
- data-split: if set and the element doesn't belong to any plugin, it's value will get split and joined by its value into the field
- **array-elements:**
  - data-split: separator (HTML allowed) between the elements, defaults to `', '`
  - data-new: value to insert into new created elements
  - data-attributes: an object to set the attributes of the child HTML-elements. tagName as special key will set the tag-name of the child HTML-elements
- **key-fields:**
  - data-trim: if `false` or `0` then the value will get saved as string else as object providing following properties: `ctrl`, `alt`, `shift`, `meta`, `code`, `char`
  - data-split: separator between different modifiers and the key-code of the value that gets saved (only takes effect if trimming)
  - data-short: if not `false` or `0` then modifier-keys get saved as first uppercase character (only takes effect if trimming)
- **select:**
  - data-options: an array of objects containing `text` and `value` attributes.

The methods of the `settings` module:

- **`registerPlugin(plugin[, types])`** Registers the given plugin and associates it to the given types if any, otherwise the plugins default types will get used.
- **`get()`** Returns the saved object.
- **`set(hash, settings[, wrapper[, callback[, notify]]])`** Refills the fields with given settings and persists them. `hash` Identifies your plugins settings. `settings` The object to save in database (settings-wrapper if you use server-side Settings Framework). `wrapper` (default: 'form') The DOM-Element that contains all fields to fill. `callback` (default: null) Gets called when done. `notify` (default: true) Whether to display saved- and fail-notifications.
- **`sync(hash[, wrapper[, callback]])`** Resets the settings to saved ones and refills the fields.

- **persist(hash[, wrapper[, callback[, notify]]])** Reads the settings from given wrapper (default: 'form') and saves them within database.

For Settings 2.0 support the methods `load` and `save` are still available but not recommended.

Client-Side Example Template

An example template-file to use the same settings we already used server-side:

```
<h1>My Plugin</h1>
<hr />

<form id="my_form_id">
    <div class="row">
        <p>
            <h2>Settings</h2>
            A boolean: <input type="checkbox" data-key="booleans.someBool"></input>
↪<br>
            An array of checkboxes that are selected by default:
            <div data-key="booleans.moreBools" data-attributes='{"data-type":"checkbox
↪"}' data-new='true'></div><br>

            A simple input-field of any common type: <input type="password" data-key=
↪"strings.someString"></input><br>
            A simple textarea: <textarea data-key="strings.multiLineString"></
↪textarea><br>
            Array of textareas:
            <div data-key="strings.arrayOfStrings" data-attributes='{"data-type":
↪"textarea"}' data-new='Hello Kitty, ahem... World!'></div><br>

            2D-Array of numbers that persist even when empty (but not empty rows):
            <div data-key="numbers.multiArrayDimensions" data-split="<br>"
                data-attributes='{"data-type":"array","data-attributes":{"type":
↪"number"}}' data-new='[42,21]'></div><br>
            Same with persisting empty rows, but not empty numbers, if no row is␣
↪given null will get saved:
            <div data-key="numbers.multiArrayDimensions2" data-split="<br>" data-
↪empty="false"
                data-attributes='{"data-type":"array","data-empty":true,"data-
↪attributes":{"type":"number","data-empty":false}}' data-new='[42,21]'></div><br>
            Array of numbers (new: 42, step: 21):
            <div data-key="numbers.justSomeNumbers" data-attributes='{"data-type":
↪"number","step":21}' data-new='42'></div><br>
            Select with dynamic options:
            <select data-key="numbers.oneNumber" data-options='[{"value":"2","text":"2
↪"},{"value":"3","text":"3"}]'></select><br>
            Select that loads faster:
            <select data-key="numbers.anotherNumber"><br>
                <option value="2">2</option>
                <option value="3">3</option>
            </select>

            Array of Key-shortcuts (new: Ctrl+Shift+7):
            <div data-key="someKeys" data-attributes='{"data-type":"key"}' data-new=
↪'Ctrl+Shift+#55'></div><br>
        </p>
    </div>
    <button class="btn btn-lg btn-warning" id="reset">Reset</button>
```

```
    <button class="btn btn-lg btn-primary" id="save">Save</button>
</form>

<script>
    require(['settings'], function (settings) {
        var wrapper = $('#my_form_id');
        // [1]
        settings.sync('myPlugin', wrapper);
        $('#save').click(function(event) {
            event.preventDefault();
            settings.persist('myPlugin', wrapper, function(){
                socket.emit('admin.settings.syncMyPlugin');
            });
        });
        $('#reset').click(function(event) {
            event.preventDefault();
            socket.emit('admin.settings.getMyPluginDefaults', null, function (err,␣
↪data) {
                settings.set('myPlugin', data, wrapper, function(){
                    socket.emit('admin.settings.syncMyPlugin');
                });
            });
        });
      });
</script>
```

Custom Settings-Elements

If you want do define your own element-structure you can create a **plugin** for the Settings Framework.

This allows you to use a whole object like a single field which - besides comfort in using multiple similar objects - allows you to use them within arrays.

A plugin is basically an object that contains at least an attribute `types` that contains an array of strings that associate DOM-elements with your plugin.

You can add a plugin at `[1]` using the method `settings.registerPlugin`.

To customize the way the associated fields get interpreted you may add the following methods to your plugin-object:

All given elements are instances of JQuery.

All methods get called within Settings-scope.

- **`use()`** Gets called when the plugin gets registered.
- **`[HTML-Element|JQuery] create(type, tagName, data)`** Gets called when a new element should get created (eg. by expansion of an array).
- **`destruct(element)`** Gets called when the given element got removed from DOM (eg. by array-splice).
- **`init(element)`** Gets called when an element should get initialized (eg. after creation).
- **`[value] get(element, trim, empty)`** Gets called whenever the value of the given element is requested. `trim` Whether the result should get trimmed. `empty` Whether considered as empty values should get saved too.
- **`set(element, value, trim)`** Gets called whenever the value of the given element should be set to given one. `trim` Whether the value is assumed as trimmed.

For further impression take a look at the default plugins.

You should also take a look at the helper-functions within Settings in order to create your own plugins. There are a few methods that take response to call the methods of other plugins when fittingly.

Internationalising your Plugin

You are free to write your plugin in whatever language you'd like, although if you wish to support multiple languages, NodeBB has a language engine that you can utilise.

Step 1: Directory layout of translations

To begin, let's define some language keys and translations! In your plugin, create a new directory to house your translations. Keep in mind that the structure of the files *inside* this folder must match that of NodeBB itself: Each sub-directory is named after a language **key** (e.g. `en_GB`), and contains `.json` files housing the translations themselves.

```
$ cd /path/to/my/plugin
$ mkdir -p languages/en_GB
$ mkdir -p languages/es
```

In the commands above, I've created my languages folder, with two languages, English (`en_GB`), and Spanish (`es`).

Step 2: Add your translations

In the sub-directories created in Step 1, I'll create text files with a `.json` extension. These file will house the plugin's translations.

In `languages/en_GB/myplugin.json`:

```
{
    "greeting": "Hello there! How are you?"
}
```

In `languages/es/myplugin.json`:

```
{
    "greeting": "Hola! Como estás?"
}
```

Note: Remember to change the name `myplugin` to something related to your plugin!

Step 3: Tell NodeBB that you have language files to load

NodeBB won't automatically know you have translations to load, so we'll need to add a line to our `plugin.json` file to tell NodeBB to load our language files.

Open up `plugin.json` and add a new property called `languages`:

```
{
    ...
    "languages": "languages",
    ...
}
```

The value for this property is the path to wherever your language files reside, *relative to the plugin's root folder*. In this case, I've placed my languages in a folder called `languages` directly in my plugin's root folder, so I just need to put in `languages`. If my languages were in a sub-folder called `public`, then the correct value here would be `public/languages`.

Step 4: Use your translations in your plugin

There are a number of ways you can use your translations in your plugin:

Server-side

In your server-side code, you can invoke the translation engine as follows:

```
var translator = require.main.require('./public/src/modules/translator');

translator.translate('[[myplugin:greeting]]', function(translated) {
    console.log('Translated string:', translated);
});
```

Client-side

In the browser, you can invoke the translation engine as follows:

```
require(['translator'], function(translator) {
    translator.translate('[[myplugin:greeting]]', function(translated) {
        console.log('Translated string:', translated);
    });
});
```

Templates

In your templates, you don't need to do anything special to invoke the translation engine, it is run through automatically, and parses any language strings matching the following syntax: `[[resource:key]]`. So for our plugin:

```
<p>[[myplugin:greeting]]</p>
```

(Optional) Step 5: Tell NodeBB that a particular language is the default

NodeBB itself supports around 40 languages, so you couldn't possibly be expected to translate them into every language! To define a specific language as default, add the `defaultLang` property to your `plugin.json` file:

```
{
    ...
    "languages": "languages",
    "defaultLang": "es",
    ...
}
```

Now, if a user utilising a language not supported by your plugin loads a language resource for your plugin, they will see the Spanish translation, as it is the designated fallback language.

CHAPTER 9

Widgets System

Writing Widgets for NodeBB

See the original blog post for a high level overview and screenshots of the widget system.

Embedding HTML and JavaScript

You don't need to be a developer to figure this out. Head over to the Themes control panel and click on the Widgets tab. Create a new HTML widget by dragging and dropping the widget onto whatever template you want.

Copy and paste HTML or JavaScript into the widget and hit save - you're done!

You can optionally give your widget a container by dragging and dropping from the containers section onto your selected widget.

If you're looking for some sample scripts, head over to our plugins section and look for any topic labelled `nodebb-script-xyz`. Don't forget to submit your scripts and ideas as well!

Creating Widgets

You can define widgets in both plugins and themes. If you're building a plugin which simply delivers a widget (or collection of widgets), we strongly suggest you follow the `nodebb-widget-xyz` nomenclature instead when publishing.

Registering your widget

Listen to this hook to register your widget:

```
"hook": "filter:widgets.getWidgets", "method": "defineWidgets", "callbacked": true
```

Pass this back in the array:

`Content` defines the form that is displayed to customize your widget in the admin panel.

Listening to your widget

NodeBB core will call your widget on the appropriate page load by way of the hooks system. The hook will be named after your widget's namespace (see previous example) - like so: `filter:widget.render:widget_namespace`

This will pass in an object with the following useful properties:

- `obj.area` - will have `location`, `template`, `url`
- `obj.data` - will have your admin-defined data; in the example from the previous section you will be exposed an `obj.data.myKey`

Defining Widget Areas in Themes

A Widget Area is characterized by a template and a location. Themes can share widgets if they define the same Widget Areas. If an admin switches themes, widgets that were previously defined in a Widget Area incompatible with the new theme are saved.

Listen to this hook to register your Widget Area:

```
"hook": "filter:widgets.getAreas", "method": "defineWidgetAreas", "callbacked": true
```

Pass this back in the array:

```
{
        name: "Category Sidebar",
        template: "category.tpl",
        location: "sidebar"
}
```

And that's all. You can define as many Widget Areas in your theme as you wish. If you're still stuck, have a look at this commit which upgraded the Cerulean theme to use the widget system.

CHAPTER 10

Theming Engine

Creating a new NodeBB Theme

NodeBB is built on Twitter Bootstrap, which makes theming incredibly simple.

Packaging for NodeBB

NodeBB expects any installed themes to be installed via `npm`. Each individual theme is an npm package, and users can install themes through the command line, ex.:

```
npm install nodebb-theme-modern-ui
```

The theme's folder must contain at least two files for it to be a valid theme:

1. `theme.json`
2. `theme.less`

`theme.less` is where your theme's styles will reside. NodeBB expects LESS to be present in this file, and will precompile it down to CSS on-demand. For more information regarding LESS, take a look at the project homepage.

Note: A *suggested* organization for `theme.less` is to `@import` multiple smaller files instead of placing all of the styles in the main `theme.less` file.

Configuration

The theme configuration file is a simple JSON string containing all appropriate meta data regarding the theme. Please take note of the following properties:

- `id`: A unique id for a theme (e.g. "my-theme")
- `name`: A user-friendly name for the theme (e.g. "My Theme")

- `description`: A one/two line description about the theme (e.g. "This is the theme I made for my personal NodeBB")
- `screenshot`: A filename (in the same folder) that is a preview image (ideally, 370x250, or an aspect ratio of 1.48:1)
- `url`: A fully qualified URL linking back to the theme's homepage/project
- `templates`: (Optional) A system path (relative to your plugin's root directory) to the folder containing template files. If not specified, NodeBB will search for the "templates" directory, and then simply fall back to using vanilla's template files.
- `baseTheme`: (Optional) If undefined, will use nodebb-theme-persona (our current base theme) as a default for missing template files. See the Child Themes section for more details.

Child Themes

CSS / LESS

If your theme is based off of another theme, simply modify your LESS files to point to the other theme as a base, ex for topics.less:

As `topic.less` from the theme `nodebb-theme-vanilla` was imported, those styles are automatically incorporated into your theme.

Templates

You do not need to redefine all templates for your theme. If the template file does not exist in your current theme, NodeBB will inherit templates from the baseTheme that you have defined in your `theme.json` (or if undefined, it will inherit from `nodebb-theme-persona`'s templates).

If your theme is dependent on a theme that is not nodebb-theme-vanilla, you should set the `baseTheme` configuration in your `theme.json` to the appropriate theme.

Rendering Engine

How it works

Every page has an associated API call, Template file, and Language File.

For example, if you navigate to /topic/351/nodebb-wiki, the application will load three resources. The API return /api/topic/351/nodebb-wiki and the template, in this example, "topic.tpl", and the appropriate language file "topic.json"*.

Just prepend api/ to the URL's path name to discover the JSON return. Any value in that return can be utilized in your template.

*A page's name corresponds to the template and language's filename (ex. `http://domain.com/topic/xyz` correlates to `topic.tpl`).

Templating Basics

Using the API return as your guide, you can utilize any of those values in your template/logic. Using the above API call as an example, for anything in the root level of the return you can do something like:

```
{topic_name}
```

To access values in objects:

```
{privileges.read}
```

And finally you can loop through arrays and create blocks like so:

```
<!-- BEGIN posts -->
{posts.content}
<!-- END posts -->
```

The above will create X copies of the above block, for each item in the posts array.

Templating Logic

NodeBB's templating system implements some basic logic. Using the same API call as above for our example. You can write IF conditionals like so:

```
<!-- IF unreplied -->
This thread is unreplied!
<!-- ENDIF unreplied -->
```

Another example:

```
<!-- IF !disableSocialButtons -->
<button>Share on Facebook</button>
<!-- ELSE -->
Sharing has been disabled.
<!-- ENDIF !disableSocialButtons -->
```

We can check for the length of an array like so:

```
<!-- IF posts.length -->
There be some posts
<!-- ENDIF posts.length -->
```

While looping through an array, we can check if our current index is the @first or @last like so:

```
<!-- BEGIN posts -->
  <!-- IF @first -->
    <h1>Main Author: {posts.username}</h1>
  <!-- ENDIF @first -->
  {posts.content}
  <!-- IF @last -->
    End of posts. Click here to scroll to the top.
  <!-- ENDIF @last -->
<!-- END posts -->
```

For more advanced documentation, have a look at the templates.js repository

Exposing template variables to client-side JavaScript

There are two ways of letting our JS know about data from the server-side, apart from WebSockets (TODO: will be covered in a different article).

Via jQuery.get

If we require data from a different page we can make a `$.get` call to any other API call. For example, if we wanted to know more about a specific user we could make a call like so:

```
$.get(RELATIVE_PATH + '/api/user/psychobunny', {}, function(user) {
    console.log(user)
});
```

See this API call in action: http://community.nodebb.org/api/user/psychobunny

Via Template Variables

In topic.tpl for example, we can add a hidden input like so:

```
<input type="hidden" template-variable="pageCount" value="{pageCount}" />
```

The template system will immediately parse all of these and expose them via the following method:

```
ajaxify.variables.get('pageCount');
```

This is the ideal method of letting JS know about important variables within the template.

Internationalization

The template engine interfaces with the internationalization system as well. We can embed variables into language strings. Let's use this API call as well as this language file as an example. We can now do something like the following:

```
[[register:help.username_restrictions, {minimumUsernameLength},
↪{maximumUsernameLength}]]
```

Which will translate this string:

```
A unique username between %1 and %2 characters
```

to

```
A unique username between 2 and 16 characters
```

Advanced Topics

Dynamically requiring and rendering a template file from client-side JavaScript

The template engine lazy loads templates on an as-needed basis and caches them. If your code requires a template or partial on-demand then you can :

```
ajaxify.loadTemplate('myTemplate', function(myTemplate) {
        var html = templates.parse(myTemplate, myData);
});
```

You can also access the invidual blocks inside each template, which is handy for doing things like (for example) rendering a new post's `<li>` and dynamically sticking it in an already loaded `<ul>`

```
Some stuff here...
<!-- BEGIN posts -->
We just want to pull this block only.
<!-- END posts -->
... some stuff here
```

```
ajaxify.loadTemplate('myTemplate', function(myTemplate) {
        var block = templates.getBlock(myTemplate, 'posts');
        var html = templates.parse(block, myData);
});
```

Rendering templates on server-side Node.js

The templating system hooks into Express just like most other templating frameworks. Just use either `app.render` or `res.render` to parse the appropriate template.

```
res.render('myTemplate', myData);
```

```
app.render('myTemplate', myData, function(err, parsedTemplate) {
        console.log(parsedTemplate);
});
```

CHAPTER 11

Developer's Resources

Developer's Resources

Note: This section is under construction.

Core

- Building a new Admin Page (Out of date)

Plugins

- Writing plugins with Grunt
- Writing your first NodeBB plugin

Themes

Widgets

Debugging

(Linux and OSX)

- Install node-inspector from npm: `npm i node-inspector -g`
- Start NodeBB in development mode: (in NodeBB root dir): `./nodebb dev`
  - This starts up only one fork so you don't get the `EADDRINUSE` error

- In a new terminal, run `node-inspector &` (spins up as a background process, so if something goes wrong, use `ps aux | grep node-inspector` to find the pid and kill it)

This will output something like:

```
Visit http://127.0.0.1:8080/debug?ws=127.0.0.1:8080&port=5858 to start debugging.
```

- Press enter to get back to the command line
- Type `ps aux | grep app.js` to see:

```
brian           79177   0.0  0.0  2432772    660 s006  R+   10:03PM   0:00.00 grep␣
↪app.js
brian           79089   0.0  1.1  3763608 183804 s005  S+    9:41PM   0:04.91 /usr/
↪local/Cellar/node/0.12.5/bin/node app.js
```

- The pid for NodeBB is 79089 in this case, so just type the command `kill -s USR1 79089` and you should see this in the terminal where you spun up ./nodebb dev:

```
Starting debugger agent.
Debugger listening on port 5858
```

- Go load the address you got from node-inspector in your browser ^ (it might take a second to load)
- Set a breakpoint somewhere (that you know how to hit) and you should be up and running!
- Now you can also just ctrl-C to quit the NodeBB process. When you want to start it again:
- Start ./nodebb dev again
- Use `ps aux | grep app.js` to find the pid again
- Since node-inspector is still running in the background, you just have to send a `kill -s USR1 <pid>` to the new pid of the running nodebb instance, and refresh the browser window that had the debugger open originally

Troubleshooting

I still get the EADDRINUSE error

- Upgrade to node 0.12.5 (0.12.x should be sufficient)
- If you had already installed node-inspector and NodeBB prior to upgrading to node 0.12.x, you'll have to rebuild your node packages:
  - In your NodeBB root directory, run `npm rebuild`
  - In your node-inspector root dir, also run `npm rebuild` * If you nstalled node-inspector through npm (`npm install -g node-inspector`), go to `/usr/local/lib/node_modules/node_inspector` and run `npm rebuild`

CHAPTER 12

Helping out the NodeBB Project

Helping out the NodeBB Project

NodeBB is an open source project, and will forever remain free. Here's a number of ways you can help us, even if you aren't a programmer.

- Like and share our content on Facebook
- Follow us on Twitter and perhaps tweet **#NodeBB is most awesome forum software @NodeBB**
- Update our wiki! ;) We need everything from development/design tutorials to user friendly how-to guides.
- Tell everybody about NodeBB, including your grandma and her cats.
- Submit a pull request, or two, or three..
- Build a new theme
- Write a plugin
- Keep the link back to us on the footer of your own NodeBB :)
- Blog about us! Give the gift of SEO juice this Christmas
- Help Translate NodeBB - It's a really simple translation tool and you don't need to know how to code.
- Join our community and give us a hard time about bugs and missing features

Translating NodeBB to another language

NodeBB uses Transifex, which is a user friendly visual tool which allows any individual to translate text into a language of their choice. You don't need to be a programmer to do this, so what are you waiting for? Join the translation team now :)

Writing Documentation

These docs were written using Sphinx and published using rtfd.org.

You can edit these docs directly on GitHub, or by clicking on "View page source" on the top right of any page.

If you wish, you can clone the repository and compile the documentation yourself. Check out the Getting Started section for more info on how to accomplish the latter.

Documentation are auto-compiled and pushed to rtfd.org after every commit.

CHAPTER 13

Troubleshooting / Help

Need Help?

Frequently Asked Questions

If you experience difficulties setting up a NodeBB instance, perhaps one of the following may help.

How do I start/stop/restart NodeBB?

You can call the `./nodebb` executable to start and stop NodeBB:

```
$ ./nodebb

Welcome to NodeBB

Usage: ./nodebb {start|stop|reload|restart|log|setup|reset|upgrade|dev}

        start   Start the NodeBB server
        stop    Stops the NodeBB server
        reload  Restarts NodeBB
        restart Restarts NodeBB
        log     Opens the logging interface (useful for debugging)
        setup   Runs the NodeBB setup script
        reset   Disables all plugins, restores the default theme.
        activate        Activate a plugin on start up.
        plugins List all plugins that have been installed.
        upgrade Run NodeBB upgrade scripts, ensure packages are up-to-date
        dev     Start NodeBB in interactive development mode
```

How do I upgrade my NodeBB?

Please consult Upgrading NodeBB

I upgraded NodeBB and now X isn't working properly!

Please consult Upgrading NodeBB

I installed an incompatible plugin, and now my forum won't start!

If you know which plugin caused problems, disable it by running: `./nodebb reset -r nodebb-plugin-pluginName`

Otherwise, disable all plugins by running: `./nodebb reset -p`

Is it possible to install NodeBB via FTP?

It is possible to transfer the files to your remote server using FTP, but you do require shell access to the server in order to actually "start" NodeBB. Here is a handy guide for installing NodeBB on DigitalOcean

I'm getting an "npm ERR!" error

For the most part, errors involving `npm` are due to Node.js being outdated. If you see an error similar to this one while running `npm install`:

```
npm ERR! Unsupported
npm ERR! Not compatible with your version of node/npm: connect@2.7.11
```

You'll need to update your Node.js version to 4 or higher.

To do this on Ubuntu:

```
# Using Ubuntu
curl -sL https://deb.nodesource.com/setup_4.x | sudo -E bash -
sudo apt-get install -y nodejs

# Using Debian, as root
curl -sL https://deb.nodesource.com/setup_4.x | bash -
apt-get install -y nodejs
```

If successful, running the following command should show a version higher than 0.8

```
# apt-cache policy nodejs
```

URLs on my NodeBB (or emails) still have the port number in them!

If you are using nginx <../configuring/proxies/nginx> or Apache <../configuring/proxies/apache> as a reverse proxy, you don't need the port to be shown. Simply run ./nodebb setup and specify the base URL without a port number.

Alternatively, edit the `config.json` file using your favourite text editor and change `use_port` to `false`.

The "Recently Logged In IPs" section only shows 127.0.0.1

NodeBBs running behind a proxy may have difficulties determining the original IP address that requests come from. It is important that the proxy server provides the referral IP header.

In nginx, ensure that the following line is present in your `server` block:

```
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

In addition, ensure that the `use_port` option is set to `false` in your NodeBB's `config.json`

Submit Bugs on our Issue Tracker

Before reporting bugs, please ensure that the issue has not already been filed on our tracker, or has already been resolved on our support forum. If it has not been filed, feel free to create an account on GitHub and create a new issue.

Ask the NodeBB Community

Having trouble installing NodeBB? Or did something break? Don't hesitate to join our forum and ask for help. Hopefully one day you'll be able to help others too :)

CHAPTER 14

Indices and tables

- genindex
- modindex
- search