The Nodejitsu Handbook

A gentle introduction to the art of Nodejitsu

Welcome to the Nodejitsu handbook. This document will help familiarize you with deploying your Node.js applications to the cloud while also providing detailed information about Nodejitsu's platform specific features. This is a living document which you can submit patches to @ http://github.com/nodejitsu/handbook.

Who is Nodejitsu?

We are a collective of seasoned developers who have been devoted to the Node.js project since 2009. We are community leaders have who created and contributed to hundreds of open-source Node.js projects. If you have used Node.js, you've probably used code we've help create. Check out our Github.

What Is Nodejitsu?

Nodejitsu is the Platform as A Service and Marketplace for Node.js applications. Nodejitsu allows you to seamlessly deploy your Node.js applications into the cloud with a myriad of additional features. Our platform provides a robust suite of functionality to assist in the development, management, and deployment of Node.js applications. Our deployment tools are the most user-friendly in the industry and our customer support is unparalleled.

How Can I Get Started?

So you wish to learn the ways of Nodejitsu? Excellent! Reading this sentence is the first step! Below, you will find the Table Of Contents which provides an overview of the systems which comprise Nodejitsu. We suggest starting at Deploying Applications. You can also always visit our website at http://nodejitsu.com. Good Luck!

Table of Contents

Deploying Applications

We've got three basic ways to deploy your application.

- Jitsu, The Nodejitsu Command Line Tool
- Samurai, An easy to use Web Admin
- The API, A high-level JSON API

If it is your first time deploying an application and you are eager to get started, we recommend that you try out Jitsu, our CLI tool. Jitsu has a one line installer, it's self-documenting, and you'll be able to deploy your app in seconds.

Let's start with a very basic node.js http server:

```
// requires node's http module
 var http = require('http');

 // creates a new httpServer instance
 http.createServer(function (req, res) {
   // this is the callback, or request handler for the httpServer

   // respond to the browser, write some headers so the
   // browser knows what type of content we are sending
   res.writeHead(200, {'Content-Type': 'text/html'});

   // write some content to the browser that your user will see
   res.write('<h1>hello, i know nodejitsu.</h1>')

   // close the response
   res.end();
 }).listen(80); // the server will listen on port 80
```

That's all the code you'll need for starters - name the file `server.js` (or anything else you'd like), and put it in a folder named `myapp`. We'll come back to this code in a minute - now, it's time to learn some Jitsu.

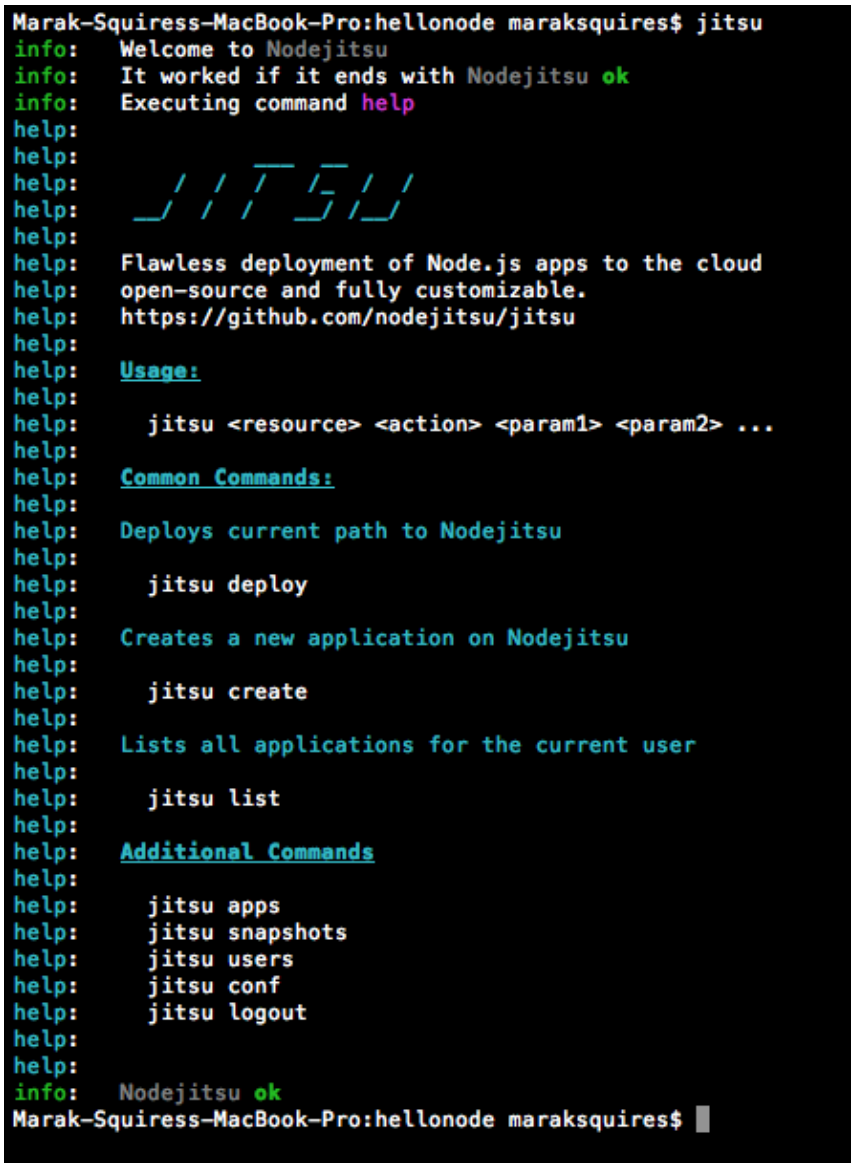
Using The Jitsu Client

Jitsu is a Command Line Interface (CLI) for interacting with the Nodejitsu platform. It's open-source and easy to use. We've designed Jitsu to be suitable for command line beginners, but still be powerful and extensible enough for production usage. If you aren't a fan of the command line or don't have terminal access you can still do everything Jitsu can do through our web admin, Samurai.

Jitsu requires the Node Package Manager (npm). If you need help installing npm go to: Installing npm

Installation

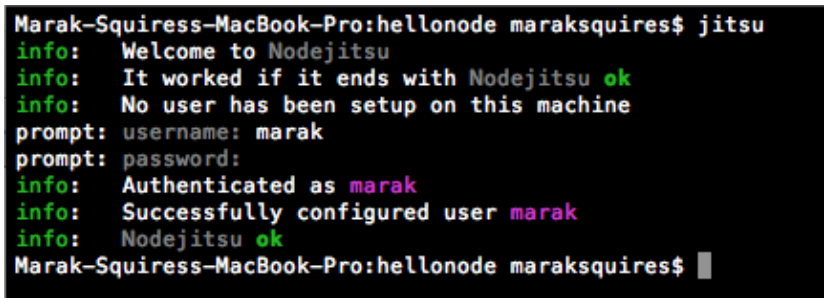
```
[sudo] npm install jitsu
```



Usage

`jitsu` is mostly self documenting. After installation, run the `jitsu` command from your command line.

If it's your first time using `jitsu`, you will be prompted to login with an existing account or create a new account.



After you've logged in, you can start deploying apps immediately!

One-line deployment

```
cd /path/to/myapp
jitsu deploy
```

This will create a new application, package.json (if you need one), and deploy the current path to Nodejitsu. If it's your first deployment, you'll be prompted for some information such as *subdomain* and *start script* but it's really easy and we promise it will only take a few seconds.

Now just open up your favorite browser, and go to yoursubdomain.nodejitsu.com. If everything has been set up correctly, then you, too, are on the path of nodejitsu!

If you have any issues deploying your node.js application please feel free to open up an issue on the Github Issues section of the jitsu homepage. We'll have someone get back to you in a flash!

Command Line Usage

`jitsu` is mostly self-documenting. Try any of these commands to get started.

Usage:

```
jitsu <resource> <action> <param1> <param2> ...
```

Common Commands:

Deploys current path to Nodejitsu

```
jitsu deploy
```

Creates a new application on Nodejitsu

```
jitsu create
```

Lists all applications for the current user

```
jitsu list
```

Additional Commands

```
jitsu apps
jitsu snapshots
jitsu users
jitsu conf
jitsu logout
```

Help

All commands will yield friendly messages to you if you specify incorrect parameters, but we have also included help commands for all available command and configuration options. If you find anything difficult to use, please open up a Github issue or pull request!

```
jitsu help
jitsu help apps
jitsu help snapshots
jitsu help users
jitsu help config
```

.jitsuconf file

All configuration data for your local `jitsu` install is located in the *.jitsuconf* file located in your home directory. Directly modifying this file is not advised. You should be able to make all configuration changes via:

```
jitsu config
```

Using The API

Nodejitsu provides a web API for developers who want to interact with the Nodejitsu platform programatically. This API is built to be RESTful and communicates via JSON. The API is the most low-level way of interacting with the Nodejitsu platform. For most deployment scenarios you should use our command line tool, jitsu, or login directly at http://nodejitsu.com

- Applications
- Snapshots
- Users
- Databases
- Logging
- Marketplace

The documentation here should be an accurate representation of our current API, but you can always look directly at our API wrappers in `jitsu` to see a working example of an application built against Nodejitsu's REST API.

Authentication

Most of the calls to the API will require that you authenticate using your Nodejitsu account. If you do not have an account it is possible to create one using the User API, the Jitsu CLI, or just by visiting http://nodejitsu.com. Currently, we support Basic Authentication. If you haven't used Basic Auth before don't fret, it's easy!

Here is an example using the command line utility, Curl.

```
// get all applications for User "Marak"
 curl --user Marak:password http://nodejitsu.com/apps/marak
```

If you are trying to use our API directly and are having issues with Basic Auth, please feel free to email support@nodejitsu.com

Applications

Applications are the core of the Nodejitsu API. Each application represents a set of Node.js code plus a package.json which contains meta-data about the application such as it's dependencies, database connections, configuration settings, authors, etc. For more information about the package.json format see: package.json

Get all Applications for a User

```
GET /apps/:user-id
```

Create a new Application

```
POST /apps/:user-id
 { package.json }
```

Start an Application

```
POST /apps/:user-id/:app-id/start
```

Stop an Application

```
POST /apps/:user-id/:app-id/stop
```

Restart an Application

```
POST /apps/:user-id/:app-id/restart
```

Update an Application

```
PUT /apps/:user-id
 { package.json }
```

Delete an Application

```
DELETE /apps/:user-id/:app-id/remove
```

Snapshots

Snapshots are an easy way to capture the current state of your application. Once a Snapshot of your application is created you can roll back and activate that Snapshot at any time.

Make an existing snapshot the active app

```
PUT /apps/:user-id/:app-id/snapshots/:id/active
```

Activate / Deploy a snapshot

```
POST /apps/:user-id/:snapshots/:id
```

Show a catalog of all Snapshot for an Application

```
GET /apps/:user-id/:app-id/snapshots
```

Show the contents of a Snapshot

```
GET /apps/:user-id/:app-id/snapshots/:id
```

Users

Create a new User / Sign-up for a free Nodejitsu account

Email address is the only required field.

```
POST /users/:user-id
 {
   email: "youremail@theinternet.com"
 }
```

Confirm a User account

All User accounts must be confirmed. When a new User is created, a confirmation email will be sent to the email address associated with that user. In this email there will be an invite code. This code must be sent to the API to confirm the account.

```
POST /users/:user-id
{
  inviteCode: "SecretCode"
}
```

Update User

```
PUT /users/:user-id
{
  password: "new_password"
}
```

Databases

Databases are an integral part of most applications. The Nodejitsu API allows you to dynamically create new hosted database instances for your applications. Cloud database hosting is provided by: CouchOne, Redis2Go and MongoHQ.

Create a new Database

```
POST /databases/:user-id/:database-id

 {
   type: "Couch || Redis || Mongo"
 }
```

Get information about a Database

```
GET /databases/:user-id/:database-id
```

Delete a Database

```
DELETE /databases/:user-id/:database-id
```

Logging

Logging is a very important feature in any professional grade Node.js application. Nodejitsu provides integrated logging solutions for your applications. Your logs are always saved and ready to be retrieved. TODO: add better description on logging

Get all logs for a user

```
GET /logs/:user-id/
```

Get logs for a specific application

```
GET /logs/:user-id/:app-id
```

Marketplace

The Marketplace is an online store where you can browse ready to deploy Node.js Applications. The Marketplace is a great place to start if you are new to Node.js development or want to share your existing Node.js Application with the world.

Get all Marketplace Applications

```
GET /marketplace
```

Get a specific Marketplace Application

```
GET /databases/:user-id/:id
```

Applications on Nodejitsu are ready to be connected to any database. If you have already have a database running, Nodejitsu can connect to your pre-existing database. If you require a new database, Nodejitsu can provide you FREE instances of several different types of databases. These free instances are great for development purposes or hobby sites. If you require a high traffic or production database we provide an easy upgrade path to industrial strength database hosting.

Creating new Databases

If you require database hosting you can create a new database instance of any of our supported databases using Samurai, Jitsu, or our API.

Existing Databases

If you already have an externally hosted Database, Nodejitsu is capable of connecting to it. We've got Database hosting if you need it, but we fully support externally hosted Databases. Feel free to drop us an email if you have any questions.

Connecting Applications to Databases

If you want to connect a Database to your Node.js application, Nodejitsu provides you with sample code for each Database type as well as the ability to specify database connection strings in your application's package.json

```
TODO: Add better package.json configuration description
```

Why open-source

A lot of Nodejitsu's technology stack is released as open-source software. We choose to do this for many reasons. Aside from being able to give back to the very awesome Node.js community, releasing pieces of our stack as open-source allows other developers to review and improve our software. We've already received invaluable contributions to our platform from developers who would have never seen our code if we had not open-sourced it.

Where to find

Nodejitsu hosts its open-source projects on Github.com. Github is website for sharing and collobrating on source code. You can get source code without creating an account and if you want to create an account it's free. You'll need a Git client if you wish to check out any of our code repositories.

You can visit our open-source project directory at: http://github.com/nodejitsu

How to contribute

Anyone can contribute to any Nodejitsu open-source projects at any time. Github has the facilities for managing patches, issues, code comments, version control, etc. If you have any questions about a project you sign up and create a Github issue. We'll make sure one our ninjas gets back to you soon.

Additional Information

If you are new to Node.js and Node.js application deployment, you might find the following section helpful.

Installing Node.js

Building and Installing Node.js

Step 1 - Pick Your Platform

Node should install out of the box on Linux, Macintosh, and Solaris.

With some effort you should be able to get it running on other Unix platforms and Windows (either via Cygwin or MinGW).

Step 2 - Prerequisites

Node has several dependencies, but fortunately most of them are distributed along with it. If you are building from source you should only need 2 things.

- **python** - version 2.4 or higher. The build tools distributed with Node run on python.
- **libssl-dev** - If you plan to use SSL/TLS encryption in your networking, you'll need this. Libssl is the library used in the openssl tool. On Linux and Unix systems it can usually be installed with your favorite package manager. The lib comes pre-installed on OS X.

Step 3a - Installing on Unix (including BSD and Mac)

Building from source

Use make to build and install Node (execute the following on the command line)

```
git clone https://github.com/joyent/node.git
cd node
export JOBS=2 # optional, sets number of parallel commands.
mkdir ~/local
./configure --prefix=$HOME/local/node
make
make install
export PATH=$HOME/local/node/bin:$PATH
```

If you have any installation problems, look at Troubleshooting Installation, try an alternate installation method, or stop into #node.js and ask questions.

Pre-built binaries

You can also install node from packages: [[Installing Node.js via package manager]]

Step 3b - Building on Windows

Pre-built binaries

Self-contained binaries are available at node-js.prcn.co.cc

Building from source

There are two ways of building Node on Windows. One is over the Cygwin emulation layer the other is using MinGW (GNU toolchain for windows). See the Cygwin and MinGW pages.

Neither builds are satisfactorily stable but it is possible to get something running.

Step 4 - Install NPM

NPM is a package manager that has become the de-facto standard for installing additional node libraries and programs. Here's the quick and easy one-liner for installing on Unix.

```
$ curl http://npmjs.org/install.sh | sh
```

To install a library e.g. jitsu (The Nodejitsu deployment CLI tool)

```
$ npm install jitsu
```

And visit https://github.com/isaacs/npm for details.

New to Node.js?

Don't be scared! There are plenty of resources out there for beginners. Here are just a few:

- The nodejs.org Official Docs:
- The #Node.js, #nodejitsu and #nodesupport rooms on irc.freenode.net
- @NodeKohai on Twitter

Understanding the package.json format

A package.json file describes your application, its dependencies, and other various application configuration. For a detailed spec on creating a package.json you can check out Isaac's fine documentation here.

Preparing a package.json for your application

Nodejitsu requires that you create a valid package.json for your application. The package.json will determine certain important pieces of information about your application which are required for deployment. Since sometimes it can get confusing when constructing your package.json file, we provide wizards in our CLI tool and on our website for creating one.

Here is an example of what your package.json might look like:

```
{
  "name": "hellonode",
  "subdomain": "hellonode",
  "scripts": {
    "start": "server.js"
  },
  "version": "0.0.0"
}
```

Notice the "scripts" property? This is where you'll store information about specific scripts in your application. The "start" property indicates the script that will get called when your application is started.

Specifying dependencies in your package.json

If your application requires additional dependencies or third-party libraries, Nodejitsu fully supports npm module dependency resolution. All you have to do is list your dependencies the exact same way you would if you were packaging a module for npm. Here is an example of the same package.json with a few dependencies.

```
{
  "name": "hellonode",
  "subdomain": "hellonode",
  "scripts": {
    "start": "server.js"
  },
  "dependencies": {
    "async": ">= 0.1.8",
```

```
    "colors": ">= 0.5.0",
    "request": ">= 1.9.0",
    "vows": ">= 0.5.8",
  },
  "version": "0.0.0"
}
```

Your dependencies will be resolved when your application deploys to Nodejitsu.