The Nodejitsu Handbook

A gentle introduction to the art of Nodejitsu

Table of Contents

Introduction

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 and about where to get support when you need it.

This is a living document which you can submit patches to at http://github.com/nodejitsu/handbook.

Who Is Nodejitsu?

We are a collection of seasoned developers who have been devoted to the Node.js community since 2009. We are community leaders who have created and contributed to hundreds of open-source Node.js projects. If you have used Node.js, you've probably used some of the projects we've helped create.

You can find a list of our open source projects at http://github.com/nodejitsu.

What Is Nodejitsu?

Nodejitsu is a Platform as a Service and a 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.

Getting Started

So you wish to learn the ways of Nodejitsu? Excellent! You only need to know three things to get started:

- We're Nodejitsu, and we can give you scalable, fault-tolerant cloud hosting for your Node.js apps - and we're the best you'll find.
- Getting started with your first app is simple with our jitsu command-line interface; we'll show you how.
- Most of our stack is open source and you can use our tools anywhere else you'd like to.

The Nodejitsu Handbook also contains information on other ways to deploy your applications, how to run your own cloud with our software, and where to get help when you need it.

Hello World: A Tutorial

In this tutorial, you will write a simple "hello world" web application in Node.js, and then deploy it using jitsu, Nodejitsu's command line interface.

Before you get started, you should have both node.js and the Node Package Manager (npm) installed.

Write A Server:

Let's start with a very basic node.js http server. Create a folder called `myapp/` and then create a file inside the folder called `server.js`. Inside this file, write the following code:

```
// 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. Save your server and get ready to deploy!

Deploy with Jitsu:

In this tutorial, we use jitsu to deploy our "hello world" application. Jitsu is a Command Line Interface for using Nodejitsu's platform. 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 the Nodejitsu Web Application.

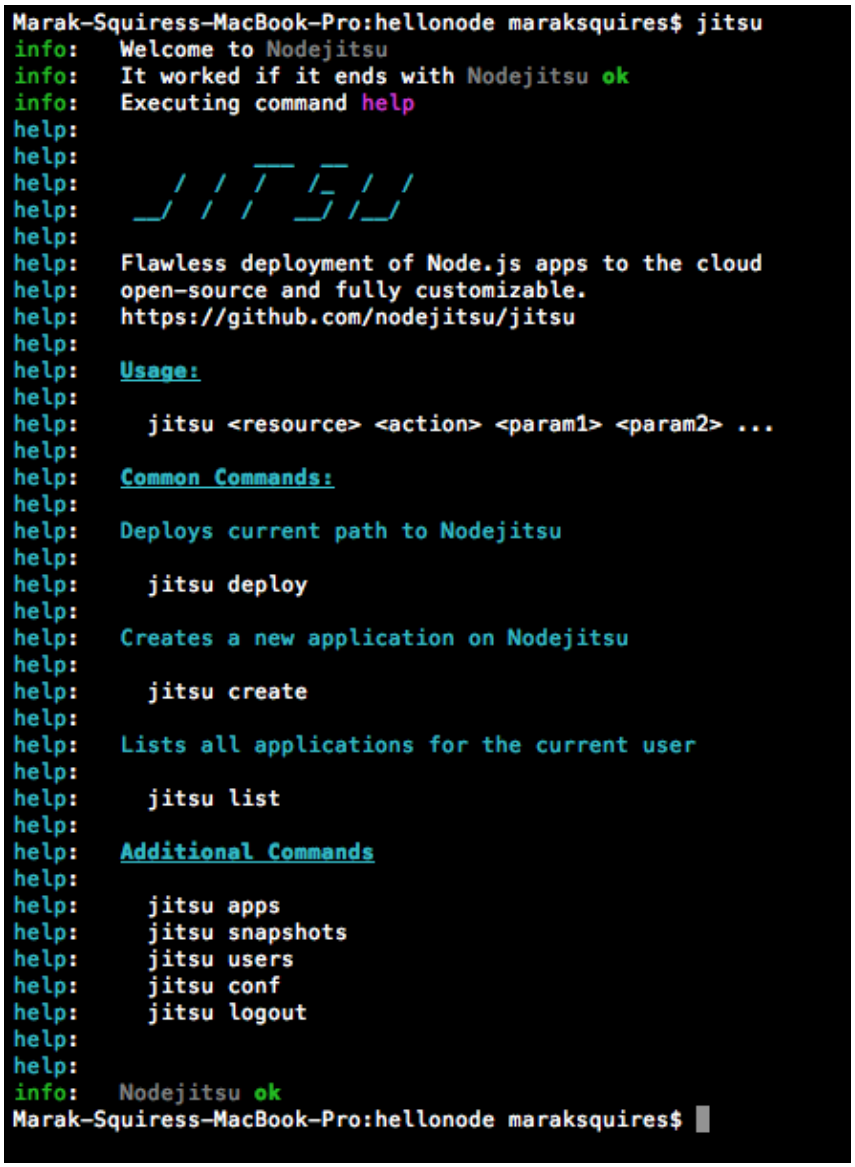
If this is your first time deploying an application and you are eager to get started, we recommend using jitsu: it has a one line installer, it's self-documenting, and with it you'll be able to deploy your app in seconds. Plus, it's what's in the tutorial.

Installation

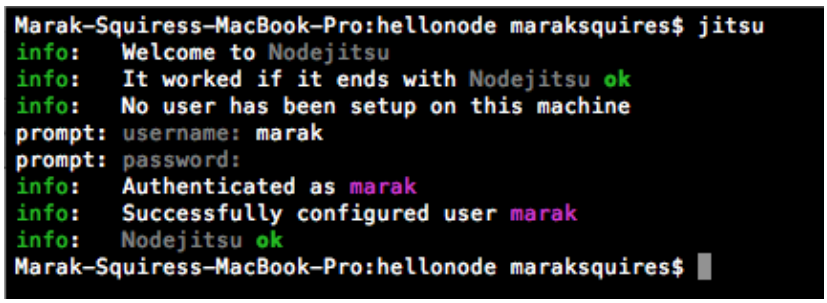
In order to install jitsu, open a terminal and type:

```
[sudo] npm install -g jitsu
```

This command will install jitsu on your system; the `-g` makes npm install it globally.



After installation, run the `jitsu` command from your command line. Since it's your first time using jitsu, you will be prompted to login with an existing account or to create a new account.



Once you've logged in, you can deploy your app immediately.

One Line Deployment

Open a terminal:

```
cd /home/me/myapp
```

```
jitsu deploy
```

This will create a new application snapshot, generate and/or update project metadata, and deploy the project in the current path to Nodejitsu. If it's your first deployment, you'll be prompted for some information such as *your app's name*, its *nodejitsu subdomain*, and its *start script*. It's really easy and we promise it will only take a few seconds.

```
prompt: subdomain (virtual-window): virtualwindow
prompt: scripts.start (server.js):
prompt: version (0.0.0):
```

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!

Features of the Nodejitsu Platform

The Nodejitsu platform makes writing and deploying web applications a snap! In addition to simple yet powerful tools for deployment, the Nodejitsu platform also has snapshot management, database hosting and connectivity, and a marketplace!

There are three main tools for deploying applications to Nodejitsu:

- Jitsu, The Nodejitsu command line tool
- The Nodejitsu Web Application, An easy to use web interface for managing your applications
- Nodejitsu's JSON API

Each of these tools allow developers to access the exact same functionality.

Snapshots

Every time you deploy to Nodejitsu, we automatically take a snapshot of your application. Using any of our tools, you can view and manage snapshots, or even roll back to an old snapshot when disaster strikes in your production environment.

Databases

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 jitsu, the Nodejitsu Web Application, or Nodejitsu's 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.

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.

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.

The Jitsu Client

Jitsu is a Command Line Interface (CLI) for interacting with the Nodejitsu platform. It's open-source and easy to use.

Installation

Jitsu is distrubited using the Node Package Manager (npm). Installing jitsu with npm is a snap:

```
[sudo] npm install -g jitsu
```

This command installs jitsu to your Node path, so that it may be run like any other global shell command.

Usage

Commands for jitsu follow this pattern:

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

For example, in `jitsu apps deploy`, "apps" is the resource and "deploy" is the action.

jitsu deploy (jitsu apps deploy)

`jitsu deploy` will attempt to deploy the application in the current directory to Nodejitsu. It deploys your application using the following steps:

1. Creates the application (if necessary)
2. Creates or validates the package.json
3. Packages and creates a new snapshot
4. Stops the application (if neccessary)
5. Starts the application

jitsu create (jitsu apps create)

`jitsu create` will create a new application. This entails generating a package.json for your app, for the purposes of deployment.

jitsu list (jitsu apps list)

`jitsu list` lists your applications, as well as their respective states, subdomains, entry points and latest snapshots.

jitsu help *resource action*

Jitsu is self-documenting. All commands will yield friendly messages to you if you specify incorrect parameters. Additionally, `jitsu help` will return useful help messages about any given resource or resource/action pair. for instance:

```
josh@pidgey:~$ jitsu help apps deploy
info:   Welcome to Nodejitsu
info:   It worked if it ends with Nodejitsu ok
info:   Executing command help apps deploy
help:
help:
help:   Deploys an application using the following steps:
help:
help:     1. Creates the application (if necessary)
help:     2. Creates or validates the package.json
help:     3. Packages and creates a new snapshot
help:     4. Stops the application (if neccessary)
help:     5. Starts the application
help:
help:   jitsu deploy
help:   jitsu apps deploy
help:
info:   Nodejitsu ok
josh@pidgey:~$
```

If no resource and/or action are specified, then `jitsu help` alone will describe what resources are available.

jitsu apps *action*

In addition to the commands aliased to `jitsu create`, `jitsu deploy` and `jitsu list`, the `apps` resource allows you to create, destroy, stop, start and otherwise interact with your applications.

jitsu config *action*

`jitsu config` commands allow you to edit your local jitsu confuration file.

jitsu snapshots *action*

`jitsu snapshots *` commands allow you to work with snapshots for your Applications on Nodejitsu. Snapshots are images of your Application's code that are deployed to the Nodejitsu Platform.

For commands that take a `<name>` parameter, if no parameter is supplied, jitsu will attempt to read the package.json from the current directory.

jitsu users *action*

`jitsu users *` commands allow you to work with new or existing Nodejitsu user accounts. You will be prompted for additional user information as required.

.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 using `jitsu config`.

Nodejitsu Web Application

The Nodejitsu Web Application allows developers to administrate their applications through a web interface. This interface allows access to all the same functionality that can be found in jitsu or the JSON API, including deployment, snapshots and database connectivity.

The web admin interface may be found at http://develop.nodejitsu.com.

JSON 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 the online administrative interface.

Jitsu is implemented by wrapping the JSON 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
```

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

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.

Get all logs for a user

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

Get logs for a specific application

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

Marketplace

Get all Marketplace Applications

```
GET /marketplace
```

Get a specific Marketplace Application

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

Create Your Own Cloud With Haibu

Haibu is an open-source tool for spawning and managing several node.js applications on a single server. It's an integral part of Nodejitsu's production stack and is fully supported by a dedicated team of core node.js developers.

By installing haibu, a user creates a development environment for themselves that mirrors the functionality of Nodejitsu's cloud platform! Any project that can be deployed on Nodejitsu can be ran by haibu.

Haibu, which is Japanese for "hive", wraps node.js applications in a "carapace" and converts them into managed "drones". This approach allows haibu to directly interact with node.js applications and add all sorts of additional functionality. Haibu also contains a plugin system, so you can easily add even more functionality.

Haibu builds on this concept of "drones" and exposes a robust and granular API for interacting with your node.js applications. At a low level, haibu's API is exposed as a RESTFul HTTP webservice. Any system that supports basic HTTP requests can communicate with a haibu server. If you are working in Node.js, haibu comes with a high-level Node.js API client.

Installation

```
[sudo] npm install -g haibu
```

This will install haibu globally.

Usage

Haibu comes with three applications, one of which is optional:

- `haibu-server` is the program that manages your node.js web applications. Haibu-server allows you to manage and track your drones.
- `haibu` is the user interface for interacting with (and administrating) a running haibu-server.
- `haibu-balancer` [*optional*] is a load balancer tool, used to split requests across multiple drones of the same application. It is entirely optional, and many deployments won't have a need for it.

It may be nice to flesh this out with an example deployment, but I think this should be relatively low priority.

Additional Documentation

For more documentation, visit haibu's github page.

Why Open Source

Most 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 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 Our Projects

Nodejitsu hosts its open-source projects on Github at http://github.com/nodejitsu. Github is a web site for sharing and collobrating on source code using git, a popular version control system. You can get our source code without creating an account at github, and if you want to create an account it's free. You will need a git client if you wish to clone any of our code repositories.

How To Contribute

Anyone can contribute to any of our Nodejitsu open-source projects at any time. Our github site has the facilities for managing patches, issues, code comments, version control, and just about anything else an open source developer could need.

Troubleshooting

This section will describe how to troubleshoot various problems.

Support

Nodejitsu has a team of developers standing by to assist users with any issues they may come across while deploying and administrating their web applications on the Nodejitsu platform. Nodejitsu strives to have a lightning-fast turnaround on all issues you may have.

E-mail

If you have any issues, feel free to email us at support@nodejitsu.com. One of our ninjas will get back to you at ninja speed!

Github

Each of Nodejitsu's open source projects uses Github Issues to manage bugs and related software problems. For example, if a user has difficulty with jitsu, they can submit an issue at https://github.com/nodejitsu/jitsu/issues. Github Issues is a great way to report bugs in our software!

IRC

Nodejitsu has a channel on <irc://freenode.net/#nodejitsu> where Nodejitsu developers are standing by to support users around the clock. Drop by to ask questions, get assistance or even just to hang out!

Twitter

Nodejitsu has an official twitter account at https://twitter.com/nodejitsu. Follow us to get the latest news about Nodejitsu, or mention us if you have issues!

Appendix: package.json

Understanding the package.json format

A package.json file describes your application, its dependencies, and other various application metadata. 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.

Appendix: Resources

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 Wiki
- The #Node.js, #nodejitsu and #nodesupport rooms on irc.freenode.net
- @NodeKohai on Twitter
- http://search.npmjs.org

Dependencies

The build process for the handbook has a few dependencies:

- make
- ronn
- htmldoc

Make and htmldoc should be available via your operating system's package manager (ie. apt-get). ronn is available on rubygems, which in turn should be available via your operating system's package manager as well. On Debian-based systems, the rubygems package does not add its bin folder (`/var/lib/gems/1.8/bin`) to your $PATH variable, so add something like:

```
PATH="/var/lib/gems/1.8/bin:$PATH"
```

to the end of your `~/.profile` file and activate it by running `. ~/.profile`.

Build Process

Once you've installed the dependencies, all you have to do is:

```
$ make
```

and the files `./book.md`, `book.pdf`, `book.html`, `API.md` and `ReadMe.md` should all be generated.