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 help 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 our open source projects at https://github.com/nodejitsu, https://github.com/flatiron, https://github.com/hookio, and https://github.com/nodeapps.

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 and where to find support.

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 node.js installed. If you are using a node.js version older than v0.6.0 (not recommended) you will also need to separately install Node Package Manager (npm).

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 will 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 app deployments 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

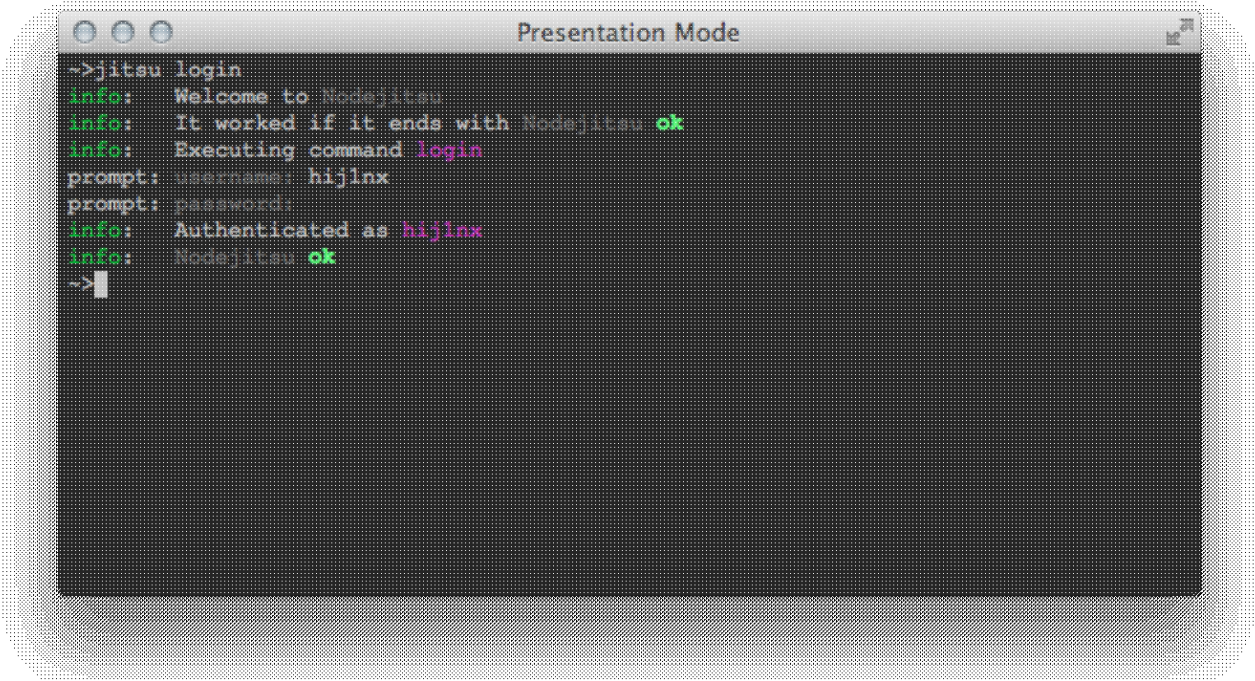In order to install jitsu, open a terminal and type:

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

This command will install jitsu on your system; the `-g` makes npm install it globally, rather than as a local module.

After installation, run the jitsu command from your command line. Since it's your first time using jitsu, you will be prompted to log in 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 (myapp): myapp
prompt: scripts.start (server.js):
prompt: version (0.0.0):
```

Now just open up your favorite browser, and go to myapp.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 more!

There are three main tools for deploying and managing 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 same functionality.

Application Scalability with Drones

Each deployed application runs as a drone on a haibu application server. Because of this architecture, one app can be served by *any amount of drones on arbitrary machines*, giving you many options for scaling your application.

Zero Downtime Deploys

When deploying a new application, nodejitsu keeps hosting your old app version until the new deploy is confirmed to be running. This means your applications never go down, even if you have a bad deploy.

Multi-Version Node Support

Nodejitsu allows users to choose which version of node they want their application to run on. In order to set your node version, specify it in your `package.json`'s "engines" field. For example:

```
{
  "author": "Nodejitsu <josh@nodejitsu.com>",
  "version": "0.1.0",
  "scripts": {
    "start": "node bin/server"
  },
  "analyze": false,
  "name": "helloworld",
  "engines": {
    "node": "v0.6.x"
  }
}
```

If no node engine is specified, jitsu will prompt for it automatically.

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. *During a deploy, nodejitsu will create a new snapshot automatically.*

Jitsu commands for snapshot management include:

- `jitsu snapshots list <app-name>` will list all snapshots for an application.
- `jitsu snapshots activate <app-name>` allows you to choose which snapshot your drones are running.
- `jitsu snapshots fetch <app-name>` will download a specified snapshot of your application to your computer.

Databases

Applications on Nodejitsu are ready to be connected to any database. If you 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 or Nodejitsu's API. Cloud database hosting is currently provided by IrisCouch, RedisToGo and MongoHQ.

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

Whenever you create a database using Nodejitsu, you will be provided with all the information you need to connect to your database.

CouchDB

If you run `jitsu databases create couchdb myCouch`, jitsu will tell you the url for your new couchdb from iriscouch:

```
info:    Welcome to Nodejitsu user
info:    It worked if it ends with Nodejitsu ok
info:    Executing command databases create couchdb myCouch
info:    Database myCouch was created.
data:    Database Type: couch
data:    Database Name: myCouch
data:    Connection url: http://subdomain.iriscouch.com:5984
data:    SSL connection url: https://subdomain.iriscouch.com:6984
info:    Nodejitsu ok
```

You can connect to this database using any http client, or a couchdb specific library. For example, you can connect with curl:

```
$ curl http://subdomain.iriscouch.com:5984
```

Or, you can connect with nano:

```
var nano = require('nano')('http://nodejitsudb944957670256.iriscouch.com:5984');
```

You can also access your database in your browser by going to http://subdomain.iriscouch.com:5984/_utils .

MongoDB

If you run `jitsu databases create mongo myMongo`, jitsu will supply a connection string for your new mongo database on mongohq:

```
info:    Welcome to Nodejitsu user
info:    It worked if it ends with Nodejitsu ok
info:    Executing command databases create mongo myMongo
info:    Database myMongo was created.
info:    Database name: myMongo
info:    Database type: mongo
info:    Connection url: mongodb://nodejitsu:pass@subdomain.mongohq.com:10057/somedatabase
info:    Nodejitsu ok
```

You can connect to this using the `mongo` CLI client tool like so:

```
$ mongo subdomain.mongohq.com:100027/somedatabase -u nodejitsu -p pass
```

or with the `mongodb-native` module:

```
var mongodb = require('mongodb');
var db = new mongodb.Db('somedatabase',
```

```
  new mongodb.Server('subdomain.mongohq.com', 10027, {})
);
db.open(function (err, db_p) {
  if (err) { throw err; }
  db.authenticate('nodejitsu', 'pass', function (err, replies) {
    // You are now connected and authenticated.
  });
});
```

or with mongoose:

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://nodejitsu:pass@subdomain.mongohq.com:10057/somedatabase');
```

You can copy-paste this url directly into your mongo library's connect method. For example, in Mongoose:

```
var mongoose = require('mongoose');
mongoose.connect("mongodb://nodejitsu:pass@staff.mongohq.com:10057/");
```

Redis

Running `jitsu databases create redis myRedis` will create a redis instance supplied by redistogo:

```
info:    Welcome to Nodejitsu user
info:     It worked if it ends with Nodejitsu ok
info:     Executing command databases create r testRedis
info:     A new redis has been created
data:    Database Type: redis
data:    Database Name: testRedis
data:    Connection host: subdomain.redistogo.com
data:    Connection port: 5309
data:    Connection auth: pass
info:    Nodejitsu ok
```

Note: Some versions of jitsu may show a connection string, eg.
`redis://nodejitsu:pass@subdomain.redistogo.com:5309`.

You can connect to your redis with the `redis-cli` cli client:

```
$ redis-cli -h subdomain.redistogo.com -p 5309 -a pass
```

or with the `redis` module:

```
var redis = require('redis');
var client = redis.createClient('subdomain.redistogo.com, 5309);
client.auth('pass', function (err) {
  if (err) { throw err; }
  // You are now authed with your redis.
});
```

Environment Variable Management

Nodejitsu allows users to modify the environment variables exposed to their apps using jitsu and our other tools. When an environment variable is changed it is necessary restart your app for it to take effect.

Available commands are `list`, `get`, `set`, `delete`, and `clear`.

`jitsu env list` will list any and all environment variables in an apps current working directory (Note: the app needs to have been deployed before the environment variables can be accessed).
`jitsu env list <myapp>` will list any and all environment variables related to `<myapp>` in an account.
`jitsu env get <key>` will display the apps key environment variable `<value>`.
`jitsu env set <key> <value>` will set the apps `<key>` environment variable to `<value>`.
`jitsu env delete <key>` will delete the apps `<key>` environment variable.
`jitsu env clear` will delete all of the apps environment variables after a prompt.

An Example:

```
$ jitsu env set NODE_ENV production
```

This will set the environment variable $NODE_ENV to have the string value "production". Remember, this will not take effect until the app is restarted (`jitsu apps restart`).

SSL on nodejitsu.com subdomains

Our balancers can proxy https to http, so you get SSL on nodejitsu.com subdomains automatically! For example, the app behind http://nodejitsu.com is serving http, but visiting https://nodejitsu.com works without any special action on our part.

Please note that this only works with `nodejitsu.com` (not `jitsu.com` or `jit.su`) at this time.

Custom Domains

We allow users to host their applications on custom domains by specifying their app's domains in their `package.json` and then properly configuring their DNS. If you'd like to know how, just read the instructions at http://dns.jit.su!

SSL Certificates for Custom Domains

Our balancers use SNI which allow them to receive SSL traffic from multiple domains---including yours! If, for example, you owned `mydomain.com` and wanted secure connections, all you need are your .pem and .key files!

Note: This feature is not exposed through our API or other tools at this time. If you need this feature, please contact support at `support@nodejitsu.com`.

Addons

Note: Coming soon!

Addons add functionality to your apps by extending and adding features to Nodejitsu's platform and integrating third party services. For instance, one of our addons provides powerful Mailchimp-based mailing list management.

Marketplace

Note: Coming soon!

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 distributed using the Node Package Manager (npm). Installing jitsu with npm is as simple as a single command:

```
[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 necessary)
5. Starts the application

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 env *action*

Jitsu allows you to set environment variables on your production environment. For example, Express, a popular node.js web framework, uses the `NODE_ENV` environment variable to change behavior based on whether the environment is for development or production. In the case of Express, nodejitsu has `NODE_ENV` set to "production" by default, but if one wanted to change this variable or add more for their application, `jitsu env` supplies the tools to do so.

jitsu config *action*

`jitsu config` commands allow you to edit your local jitsu configuration 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.

jitsu install

`jitsu install` is a built-in tool for downloading "starter apps" to speed up development. For example, here's how to use `jitsu install` to download a "hello world" application:

```
josh@onix:/tmp$ jitsu install helloworld
info:   Welcome to Nodejitsu jesusabdullah
info:   It worked if it ends with Nodejitsu ok
info:   Executing command install helloworld
info:   Installing helloworld locally.
warn:   Downloading packages from npm, this may take a moment...
info:   helloworld installed.
help:   You can now jitsu deploy this application
prompt: Would you like to start this application locally? (yes): no
info:   Nodejitsu ok
josh@onix:/tmp$ cd helloworld
josh@onix:/tmp/helloworld$ ls
bin  node_modules  package.json  ReadMe.md
```

Configuring jitsu

All user-level 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`.

Selected Properties of .jitsuconf

- *loglength*: Change this to modify the default length of returned logs from `jitsu logs`.
- *loglevel*: Change this to modify the logging levels displayed by jitsu. Defaults to "info".
- *colors*: Change this to false to turn off jitsu's colors. Defaults to `true`.
- *analyze*: Change this to false to disable require-analyzer for all apps. Defaults to `true`.

Nodejitsu Web Application

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

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.

API Clients

Nodejitsu has a JSON API client for node.js, which may be found here (along with API clients in other languages as they are developed). Jitsu is implemented by using the node.js API client.

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 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://api.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 and authors. 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

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

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

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

```
POST /logs/:user-id/
{
  "from": "NOW-3YEARS",
  "until": "NOW",
  "rows": 15
}
```

Get logs for a specific application

```
POST /logs/:user-id/:app-id
{
  "from": "NOW-3YEARS",
  "until": "NOW",
  "rows": 15
}
```

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 haibu -g
```

This will install haibu globally. You can also grab the source directly from git.

Usage

To start haibu, all you have to do is run `haibu`:

```
$ haibu
     __            __
    / /_   _____  __  / /_      __  __
   / __ \ / __ \/ / / / __ \   / / / /
  / / / / / /_/ / / / / / /_/ / / /_/ /
 /_/ /_/  \__,_/ /_/ /_/\__,_/  \__,_/

  This is Open Source Software available under
  the MIT License.

  Â© 2010 Nodejitsu Inc.
  All Rights Reserved - www.nodejitsu.com
  haibu started @ 10.0.1.4 on port 9002 as api-server
    using plugins: config, exceptions, directories, log, http
```

Haibu is an http server that exposes a REST api on port 9002. You can either access this API client with a regular HTTP client, or use our haibu-api module. Unfortunately, jitsu does not work with haibu's HTTP API, only the nodejitsu API.

Additional Documentation

For more documentation, visit haibu's github page.

Open Source Projects

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:

- *https://github.com/nodejitsu*: Open source components of Nodejitsu's PaaS
- *https://github.com/flatiron*: Nodejitsu's "anti-framework" for building web and cli applications
- *https://github.com/nodeapps*: A collection of node.js applications ready to use

Github is a web site for sharing and collaborating 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 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.

Frequently Asked Questions

For more information about pricing, see the pricing FAQ.

"How do I reset my password?"

One way is to use jitsu. Simply type:

```
jitsu users forgot :username
```

where `:username` is your username. Alternately, go to http://develop.nodejitsu.com/ and click the "forgot password" link, where you will be prompted for your username. Either process will send you an email with further instructions.

"Is there a cheatsheet somewhere?"

There sure is! Check out http://cheatsheet.nodejitsu.com.

"How are programs kept alive? Do I need to use Forever?"

Nodejitsu's cloud services watch your programs for you! You shouldn't have to do anything special to keep your apps running, much less use Forever.

"How can I make my app use a port other than port 80?"

Connecting to other servers using arbitrary ports requires no special considerations. However, *listening* for outside connections is currently limited to port 80 on the Nodejitsu platform because we require http host headers for domain name resolution of subdomains. Consequentially, each subdomain may only host one listening service.

The ability to host tcp applications on nodejitsu and listen on non-80 ports is on our roadmap but has no associated timeline.

"How can I turn off the require-analyzer in jitsu? I want to manage my own dependencies!"

There are three ways to disable the require-analyzer:

- Use the `--noanalyze` flag when running jitsu commands to disable it on a one-time basis.
- Add `"analyze": false` to your package.json to disable it on a per-app basis.
- Set "analyze" to `false` in your `~/.jitsuconf` to disable it on a global level.

"Why won't this C++ addon compile?"

Many C++ addons require libraries that are not included in Nodejitsu's infrastructure by default. For example, node-canvas requires cairo. Nodejitsu has cairo and many other such libraries, but may not have some more obscure ones.

"How do I specify which files not to bundle? How do I know what files are getting bundled?"

Jitsu uses npm to bundle files, meaning that jitsu bundles files in exactly the same manner than npm bundles published modules. You can read about this in npm's documentation.

In more detail: npm uses a file called `.npmignore`, which should contain a list of files and folders to ignore for the purpose of bundling. If this file does not exist, npm will use git's ignore file, called `.gitignore`, instead. This means that, if you want to bundle files that are ignored by git, you should create an `.npmignore` even if it's blank.

Finally, jitsu has the ability to bundle your app without deploying with the `jitsu package create` command. You can use this to make sure that the resulting .tgz file is as you expect.

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

You can also contact us via email, at **support@nodejitsu.com**.

IRC and Kohai

Nodejitsu has a channel on freenode at irc://irc.freenode.net/#nodejitsu (http://webchat.jit.su), where Nodejitsu staff are standing by to support users around the clock. Drop by to ask questions, get assistance or even just to hang out!

Kohai is an IRC bot that has some basic abilities to help you among other features. To bring up his help dialogue just type `!help` into the #nodejitsu channel and Kohai will message you.

Github Issues

Each of Nodejitsu's open source projects uses Github Issues to manage bugs and related software problems. Github Issues is most useful for developers. For example, if a developer finds a bug in our open-source http proxy, they can submit an issue at https://github.com/nodejitsu/node-http-proxy/issues and tell us about their bug.

Table of Appendices

- package.json
- Resources
- Building The Handbook

Appendix: package.json

package.json.jit.su is an interactive package.json properties explorer! Highly recommended.

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": "node 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. Usage is compatible with `npm start`.

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.x",
    "colors": "0.5.x",
    "request": "1.9.x"
  },
  "version": "0.0.0"
}
```

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

Nodejitsu-Specific package.json Properties

A few package.json properties have special behavior on the Nodejitsu platform:

- *subdomain*: Specify the subdomain for your hosted app's nodejitsu url (for example, `subdomain.nodejitsu.com`.
- *domains*: A list of custom domains for your hosted app. See http://dns.nodejitsu.com.
- *env*: Specify environment variables for your app (for example, NODE_ENV="production" is set by default).
- *scripts.start*: This field is also used for `npm start`. However, nodejitsu's current implementation takes a path, whereas npm's implementation takes a shell command.
- *analyze*: Set this to "false" to force jitsu to not analyze for the app's dependencies.

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 document all of Node.js's core APIs.
- The Node.js Wiki contains information such as an FAQ, installation instructions, and lists of modules.
- The #Node.js and #nodejitsu channels on irc.freenode.net are ready to help you with any Node.js issues and concerns you may have.
- @NodeKohai on Twitter is an irc bot that shares Node.js tweets with the #nodejitsu irc channel. Ask it a quick question and it just might give you an answer!
- http://npmjs.org is a great place to check for modules that might already solve your problem.
- http://cheatsheet.jit.su is a nodejitsu cheatsheet listing the most common commands.
- http://package.json.jit.su is an interactive reference for the package.json format.
- http://blog.nodejitsu.com/npm-cheatsheet is a quick cheatsheet for npm commands.

Appendix: Building the Nodejitsu Handbook

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.