Cloudnode User's Guide
Version 1.0

Hans J Schroeder
http://cloudno.de/

October 4, 2011

Contents

Chapter 1

Quick Start Guide

If this is your first time using Cloudnode, be sure to first install the command line client and look over the platform prerequisites before proceeding. Also, make sure you have a working Node installation, otherwise any node commands listed below will not work.

- Create a new cloudnode app
- Initialize your local working copy
- Commit to your new app and deploy

1.1 Create a new cloudnode app

Go to My Apps[1], click on "New Application" and fill in the form. This will actually:

- Create a new sub-domain for your application
- Provision a virtual machine for your application
- Setup a git repository to hold your application code

1.2 Initialize your local working copy

The easiest way is to use the command line client[2] to setup your local working copy.

```
$ cloudnode app init <app name>
cloudnode info initializing git repo for <app name> into folder <app name>
cloudnode info cloning the repo git clone cloudnode@git.cloudno.de:/git/hs/62-6fc0d44abf9974
cloudnode info clone complete
cloudnode info writing app data to config
cloudnode info writing app files
cloudnode info processing the initial commit
```

[1]https://cloudno.de/myapps
[2]/cloudnode-command-line

```
cloudnode info attemping to start the new app.
cloudnode info hello started.
cloudnode info Some helpful app commands:

 cd ./<app name>
 curl http://<app name>.cloudno.de/
  cloudnode app info
  cloudnode app logs
  cloudnode app stop|start|restart
```

This single command automates the following steps, which could also have been executed instead:

1. Create a local sub directory for the application
2. Initialize a local repository (git init)
3. Clone the remote repo and the the remote origin URL (git clone . . .)
4. Create a sample server.js file with a simple “Hello World” application
5. Add the new file to the local repository (git add .)
6. Commit the changes (git commit -am “Initial commit”)
7. Push the changes live (git push origin master)

You can also launch the application locally to see if it works by running **node server.js** and navigating to [[http://127.0.0.1:8080/in a web browser. Use any port number and IP address; they will be transparently overridden when your app is deployed on Cloudnode.

1.3 Commit to your new app and deploy

To commit changes to your application use the normal git workflow:

- Edit your files (vi server.js)
- Commit the changes (git commit -am “Change log”)
- Push the changes online (git push origin master)

Commit the changes

```
$ git commit -am "Message changed"
Created commit f57b7a0: Message changed
1 files changed, 1 insertions(+), 1 deletions(-)
```

Push the changes online

```
$ git push origin master
Counting objects: 5, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 315 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To cloudnode@git.cloudno.de:/git/hs/62-6fc0d44abf9974b91625cc10ff118871.git
 5f3a0f9..f57b7a0  master -> master
Syncing repo with your node VM
From /git/hs/62-6fc0d44abf9974b91625cc10ff118871.git/.
 5f3a0f9..f57b7a0  master     -> origin/master
Updating 5f3a0f9..f57b7a0
Fast forward
 server.js |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

====  Compiling hello...
====  Restarting your app: hello
====  App restarted
====  App successfully deployed to http://hello.cloudno.de

  Finished.
```

Now your new version is deployed on Cloudnode and the application has been restarted. You can see it running in your browser at the .cloudno.de URL listed in the push output.

Chapter 2

Prerequisites

Since Cloudnode is a cloud hosting environment, your app must conform to a couple different requirements and prerequisites in order to run on it. By following these best practices in app architecture and design, you can ensure that your app will run well on the Cloudnode platform.

- Required Software
- Best Practices
- Read-only Filesystem
- Logs
- Dependency Management
- App Initialization
- Git

2.1 Required Software

The workflow used to develop and deploy apps on Cloudnode depends on Git, Node.JS, npm and our command line client, which in fact is a node app itself. The tools are available for all major OS'es. See the following links

- git - http://git-scm.com/ - The distributed version control sysstem
- node.js - http://nodejs.org - The node server
- npm - http://npmjs.org/ - The Node Package Manager
- cloudnode - **npm install cloudnode-cli** - The Cloudnode command line

2.2 Best Practices

Take care to keep the size of your app small.

Do not include large files (documents, media files, data files, etc.) in your app. These should be hosted on an outside asset store such as Amazon S3.

Only include modules that you will use.

2.3 Read-only Filesystem

Cloudnode apps are not able to write to file filesystem in general. However there are a couple special locations to which you can write files.

Cloud storage (/mnt)

All Cloudnode apps have access to limited persistent file storage under /mnt. Cloud storage is persistent between requests and app instances.

Cloud storage is designed to store files generated from within your app (e.g. generated stylesheets or asset packages). See Cloud Storage[1] for details.

Temporary files (/tmp)

The /tmp directory in your app root is writeable (for files of reasonable size), but anything written to it should not be expected to endure over hours.

2.4 Logs

Logs on the Cloudnode platform should be considered transient. It is possible to tail the last 100 lines through the command line.

See Troubleshooting[2] for full details.

2.5 Dependency Management

If your app depends on any npm packages, they need to be installed into your Node VM. See Node package manager[3] for details.

2.6 App Initialization

Node looks for a startup file normally called **server.js** to boot your application. You can choose a different file and directory, when you create the application. For details see the Cloudnode command line[4].

[1] /cloud-storage
[2] /troubleshooting
[3] /node-package-manager
[4] /cloudnode-command-line

2.7 Git

Git[5] is the only way to push code to Cloudnode for deployment. This means your app must be part of a Git repository. You don't necessarily need to use Git for version control during development, but when it comes time to deploy to Cloudnode, your app code does need to be committed to a Git repo.

Deployment is as easy as **git push origin master**. See the Quick Start Guide[6] for more details.

For additional help on using Git see the excellent help files at GitHub especially the setup and troubleshooting guides when using ssh key and key phrases: http://help.github.com/

Git Submodules

Git submodules are supported on the Cloudnode platform.

[5]http://git-scm.com
[6]/quick-start-guide

Chapter 3

Support Resources

If you can't find the information here you are looking for, use the Cloudnode support group[1] to contact us or the other users.

You can also use the group for technical questions or to showcase your apps.

[1] http://support.cloudno.de

Chapter 4

Cloudnode command line

Current version: 0.2.20 (08/28/11) (see also: how to update, changelog)

The Cloudnode command line tool is an interface to the Cloudnode Web API[1] and includes support for creating apps, reading log files, managing apps and user accounts, setting up domains, and configuring apps.

The command line tool is itself a Node.js application the runs on the client side. See the Prerequisites[2] for details.

4.1 Installation

If you haven't already done, sign up for a Cloudnode account. You will need to enter your credentials into the command line tool to authorize most of the requests.

The command line tool can be installed using npm[3] which also takes care to resolve dependencies.

```
$ npm install cloudnode-cli
```

4.2 Usage

After having installed the client, enter your credentials. This needs to be done only once. The user name is the same as in your OpenID, the password is the API key shown on your account page[4].

```
$ cloudnode user setup <username> <password>

cloudnode info verifying credentials
cloudnode info user verified..
```

[1] /api
[2] /prerequisites
[3] /node-package-manager
[4] https://cloudno.de/account?admin

```
cloudnode info writing user data to config
```

Now start the client by typing "cloudnode". It will show a list of the available commands.

```
$ cloudnode

cloudnode info showing all available sub commands
cloudnode status
cloudnode coupon
cloudnode apps
cloudnode app
cloudnode user
cloudnode appdomain
cloudnode domain
cloudnode npm
cloudnode appnpm
cloudnode info For more help, type cloudnode help <command>
```

The commands are divided into three types: general commands, user and app commands.

General Commands

The status command checks the platform status and displays the number of hosted and running applications.

```
$ cloudnode status
```

The apps command displays all your applications together with their port and running status.

```
$ cloudnode apps
```

User Commands

Type "cloudnode user" to get an overview of the commands in this category:

```
$ cloudnode user
cloudnode user register <coupon-code> - Register a user
cloudnode user setup <username> <password> - Setup this user
cloudnode user setpass <password>
               - Set a new password for this user
cloudnode user setkey </path/to/sshkey>
               - Set an sshkey (if no argument, ~/.ssh/id_rsa.pub is used)
cloudnode user create <username> <password> <email address>
```

```
                   <file containing ssh public key> <coupon code> - Create a user
```

App Commands

Type “cloudnode app” to get an overview of the commands in this category:

```
$ cloudnode app
cloudnode <appname> is not required if inside an app directory after you call setup
cloudnode app setup <appname> - Configure this app for future app commands
cloudnode app info <appname> - Returns app specific information
cloudnode app logs <appname> - Returns app logs
cloudnode app stop|start|restart <appname> - Controls app status.
cloudnode app create <appname> <startfile>
              - Creates a new app named <appname>, <startfile> is optional.
cloudnode app init <appname> - Fetches the remote repo and sets it up.
cloudnode app clone <appname> - Fetches the remote repo.
```

NPM Commands

Type “cloudnode npm” or “cloudnode appnpm” to get an overview of the commands in this category:

```
$ cloudnode npm
cloudnode All arguments after install|update|uninstall will be sent to npm as packages.
cloudnode npm install <packages> - Installs the list of specified packages to this app.
cloudnode npm update <packages> - Update the list of specified packages to this app.
cloudnode npm uninstall <packages> - Removes the list of specified packages to this app.
```

Because every node application is running in its own virtual machine, you need to install every module you application depends on.

Domain Commands

The domain command are used to setup and manage custom domains for you appications. See the Custom Domains[5] chapter for additional information on this.

[5]/custom-domains

Chapter 5

Node package manager

Current version: npm 1.0.22 (08/07/11) (see also: how to update, changelog)

Cloudnode uses npm[1] for dependency management. For an introduction and additional information see the Author's article on How To Node[2]. If you need more specific information, the best place to look is npm's help system itself, which is very extensive. Just use npm help .

When you application depends on some other modules use the Cloudnode command line to install the required packages into your VM. Additional dependencies wil be resolved by npm. If your application requires for instance express, run the following command:

```
$ cloudnode npm install express
```

All arguments after install will be sent to npm as package names. You can also use the npm command to update and uninstall packages. To get an overview of all npm commands run "cloudnode npm":

```
$ cloudnode npm
cloudnode All arguments after install|update|uninstall will be sent to npm as packages.
cloudnode npm list - Lists the installed npm packages for this app.
cloudnode npm install <packages> - Installs the list of specified packages to this app.
cloudnode npm update <packages> - Update the list of specified packages to this app.
cloudnode npm uninstall <packages> - Removes the list of specified packages to this app.
```

Remember that each application runs in its own isolated environment. You need to handle dependencies for each application individually. When you want to share program code among your applications and that code might also be useful for other node users, consider to publish your own npm package.

[1]http://npmjs.org/
[2]http://howtonode.org/introduction-to-npm

Chapter 6

API

Cloudnode is build on a RESTful API the allows to execute all commands from the web frontend, the command line tool or a future client-side app.

We are currently running the "stable" version of Node v.0.4.10 and we support Node.JS VMs, we call Cloudnode machines. This means that you can install your own NPM modules. Git is required to push updates to your Cloudnode machine. The following API calls are defined:

- Status (section 6)
- User (section 6)
- App (section 6)
- Apps (section 6)
- Env (section 6)
- NPM (section 6)
- Appdomains (**??**)
- CLI Commands (**??**)
- Git (chapter 7)

Status

Get Status :: GET Base URL: https://cloudno.de

/status - Returns platform status and number of apps running

```
$ curl https://cloudno.de/status
```

User

Register User :: POST (Coupon is required.) Base URL: https://cloudno.de

/user - creates user account (pass in user and password and email and id_rsa.pub string) Ensure that all + in the ssh key are substituted for their %2B counter parts, else your key will break. Run this on your command line to copy your RSA string and swap out the plus signs: “cat ~/.ssh/id_rsa.pub | sed s/‘+’/‘%2B’/g | pbcopy”

```
$ curl -X POST -d "user=testuser&password=123& \
  email=chris@cloudno.de&rsakey=ssh-rsa AAAAB3NzaC1yc..." https://cloudno.de/user
```

Update User :: PUT Base URL: https://api.cloudno.de

/user - update user account (pass in password and/or RSA key - “cat ~/.ssh/id_rsa.pub | sed s/‘+’/‘%2B’/g | pbcopy”)

```
$ curl -X PUT -u "testuser:123" -d "password=test" https://api.cloudno.de/user
$ curl -X PUT -u "testuser:123" -d "rsakey=1234567" https://api.cloudno.de/user
```

Delete User :: DELETE Base URL: https://api.cloudno.de

/user - delete user account (requires basic auth)

```
$ curl -X DELETE -u "testuser:123" https://api.cloudno.de/user
```

App

Create Application :: POST Base URL: https://api.cloudno.de

/app - create nodejs app for hosting (requires basic auth and returns the port address required for use along with a git repo to push to)

```
$ curl -X POST -u "testuser:123" -d "appname=a&start=hello.js" https://api.cloudno.de/app
```

Change Application :: PUT Base URL: https://api.cloudno.de

/app - update starting app name (requires basic auth, appname, and starting page and returns the port address required for use along with a git repo to push to and running status of the app)

```
$ curl -X PUT -u "testuser:123" -d "appname=a&start=hello1.js" https://api.cloudno.de/app
```

Start/Stop Application :: POST Base URL: https://api.cloudno.de

/app - start and stop your hosted nodejs app (requires basic auth, appname, and running=true|false and returns the port address required for use along with a git repo to push to)

```
$ curl -X PUT -u "testuser:123" -d "appname=a&running=true" https://api.cloudno.de/app
```

Delete Application :: DELETE Base URL: https://api.cloudno.de

/app - delete nodejs app (requires basic auth and appname)

```
$ curl -X DELETE -u "testuser:123" -d "appname=test" https://api.cloudno.de/app
```

Application Information :: GET Base URL: https://api.cloudno.de

/app/- get nodejs app info (requires basic auth and appname)

```
$ curl -u "testuser:123" https://api.cloudno.de/app/appname
```

Apps

All Applications Information :: GET Base URL: https://api.cloudno.de

/apps - get all nodejs app info(requires basic auth)

```
$ curl -u "testuser:123" https://api.cloudno.de/apps
```

Env

Create/Update Environment :: PUT Base URL: https://api.cloudno.de

/env - create/update environment key/value pair (requires basic authentication, appname and key/value)

```
$ curl -X PUT -u "testuser:123" -d "appname=test&key=color&value=red" https://api.cloudno.de
```

Delete Environment :: DELETE Base URL: https://api.cloudno.de

/env - delete environment key/value pair (requires basic authentication, appname and key/value)

```
$ curl -X DELETE -u "testuser:123" -d "appname=test&key=color" https://api.cloudno.de/env
```

Get Environment :: GET Base URL: https://api.cloudno.de

/env - get all environment key/value pairs (requires basic authentication and appname)

```
$ curl -u "testuser:123" https://api.cloudno.de/env/test
```

NPM

Install/Upgrade/Uninstall NPM Packages :: POST Base URL: https://api.cloudno.de

/npm - Allows you to manage the NPM packages for an application.

```
$ curl -X POST -u "testuser:123" -d "appname=a&action=install&package=express" \
       https://api.cloudno.de/npm

$ curl -X POST -u "testuser:123" -d "appname=a&action=update&package=express" \
       https://api.cloudno.de/npm

$ curl -X POST -u "testuser:123" -d "appname=a&action=uninstall&package=express" \
       https://api.cloudno.de/npm
```

Appdomains - Add DNS A Record

Create Application Domain :: POST Base URL: https://api.cloudno.de

/appdomains - create app domain for hosting example.com (requires basic auth)

```
$ curl -X POST -u "testuser:123" -d "appname=test&domain=example.com" \
       https://api.cloudno.de/appdomains
```

Delete Application Domain :: DELETE Base URL: https://api.cloudno.de

/appdomains - delete app domain for hosting example.com (requires basic auth)

```
$ curl -X DELETE -u "testuser:123" -d "appname=test&domain=example.com" \
       https://api.cloudno.de/appdomains
```

Application Domain Information :: GET Base URL: https://api.cloudno.de

/appdomains - get list of your domains (requires basic auth)

```
$ curl -u "testuser:123" https://api.cloudno.de/appdomains
```

CLI Commands

You can install our Command Line Interface by running “npm install cloudnode-cli”

```
cloudnode <command> <param1> <param2>
```

Commands are:

```
$ cloudnode coupon <email address>
$ cloudnode user create <username> <password> <email address> \
            <file containing ssh public key> <coupon code>
$ cloudnode user setup <username> <password>
```

The commands below require you to have run ‘user setup’ before:

```
$ cloudnode user setpass <new password>
```

You should run user setup after running setpass.

```
$ cloudnode user setkey <file containing ssh public key>
$ cloudnode apps list
$ cloudnode app create <app-name> <initial js file>
$ cloudnode app info <app-name>
$ cloudnode app logs <app-name>
$ cloudnode app start <app-name>
$ cloudnode app restart <app-name>
$ cloudnode app stop <app-name>
$ cloudnode app gitreset <app-name>
$ cloudnode npm install <app-name> <package name>
$ cloudnode npm upgrade <app-name> <package name>
$ cloudnode npm uninstall <app-name> <package name>
$ cloudnode appdomain add <app-name> <domain-name>
$ cloudnode appdomain delete <app-name> <domain-name>
$ cloudnode appdomains
```

Git

Deploying and updating your Node.js application is simple.

```
$ curl -X POST -u "testuser:123" -d "appname=myapp&start=hello.js" \
            https://api.cloudno.de/app
```

Upon creating or changing your application via our API, you will receive a Git reop URL from our API response. Add a Cloudnode remote to your project as follows:

```
     $ git remote add cloudnode the_url_returned_by_our_api
```

Finally push your updates to your new Cloudnode environment as follows:

```
     $ git push cloudnode master
```

Start your application.

```
     $ curl -X PUT -u "testuser:123" -d "appname=myapp&running=true" \
            https://api.cloudno.de/app
```

Visit your application via http://myapp.cloudno.de

Chapter 7

Git

Git is a distributed version control system. Each application on Cloudnode has its own repository. On every push command the Node VM is updated with the latest version of your application. An automatic restart ensures that the most recent version goes live after the push command completes.

Cloudnode Remote Branches Special Directories Submodules

Cloudnode Remote

When creating a new Cloudnode App with the command line client, a Git remote pointing to Cloudnode is automatically configured. If you wish to configure the remote manually or re-add the remote under a different name, you can:

```
$ git remote add cloudnode git@cloudno.de:demoapp.git
```

In this example Cloudnode (the third argument to git) is the name of the remote and demoapp should be replaced with your app name.

Branches

Cloudnode will ignore branches other than master if they are pushed. Only the master branch is used for deployment.

You can, of course, push any local branch to the master branch of the Cloudnode remote. The syntax for that is:

```
$ git push cloudnode demobranch:master
```

In this example Cloudnode (the second argument to git) is the name of the remote and demobranch is the name of the local branch being deployed to Cloudnode.

Special Directories

A couple directories are handled specially by the Cloudnode app compiler when receiving your app code. The following directories will be ignored if they exist in your Git repository, so they can be used by the platform for special purposes:

mnt/Used for mounting your app's Cloud Storage. log/Used for log collection. See Logging for details and usage. tmp/Writeable directory available for transient file storage. See Platform Prerequisites for details and usage.

Submodules

Git submodules are supported on the Cloudnode platform.

Chapter 8

Troubleshooting

If you need help with your login or with creating /updating your application, please see the following guides:

- Login to the Cloudnode web site
- Using the API /CLI
- SSH issues
- Analysing the application log files

8.1 Login to the Cloudnode web site

With Cloudnode you don't need to remember passwords, nor you need to worry about leaked or hacked passwords, because we simply don't use passwords and rely on OpenID /OAuth instead. Use your existing account at one of the supported providers to login into our web site.

8.2 Using the API /CLI

The API uses basic authentication over a secure SSL connection. We plan to change this as soon as the Nodester platform supports OAuth. The user name is the same as in your OpenID. Instead of a password an API key is used. The API key is maintained by the platform and is shown on your account[1] page. **You need to create at least one application to get access to your API key.**

To test the platform and the physical connection, use the status command:

```
$ cloudnode status
cloudnode info checking api status for: api.cloudno.de
cloudnode info using secure connection
cloudnode info status up
cloudnode info appshosted 23
```

[1] https://cloudno.de/account?admin

```
cloudnode info appsrunning 13
```

This ensures, that the platform is up and running and that you can open a secure connection to it.

The next step is to check your credentials by running a command, that requires authentication.

```
$ cloudnode apps
cloudnode info hello on port 8062 running: empty
```

This command will list all your applications. If you get an ERROR instead, setup the cloudnode client with your user name and API key as a password.

```
$ cloudnode user setup <user name> <api key>
cloudnode info verifying credentials
cloudnode info user verified..
cloudnode info writing user data to config
```

You are now ready to use the command line tool.

8.3 SSH issues

Make sure that you have uploaded your public SSH key to your account[2]. Compare the fingerprint that is displayed on your account page with the one displayed when running the following command on your local computer:

```
$ ssh-keygen -l -f .ssh/id_rsa.pub

2048 ab:f2:46:70:30:48:31:05:79:e2:eb:f5:95:38:08:50 .ssh/id_rsa.pub
```

With your SSH key on file, you can create your first application. The ssh connection will not work, until you have created an application.

After your first app is created, the next step is testing your connection by running *ssh cloudnode@git.cloudno.de*. If your key works, you should get a success message:

```
$ ssh cloudnode@git.cloudno.de

Hi <user>! You've successfully authenticated, but Cloudnode does not provide shell access.
Connection to git.cloudno.de closed.
```

[2]https://cloudno.de/account?ssh

8.4 Permission denied (publickey)

This is usually caused when ssh cannot find your keys. Make sure your key is in the default location, *~/.ssh*. If you run ssh-keygen again and just press enter at all 3 prompts it will be placed here automatically. Then you can add the contents of id_rsa.pub to your account.

8.5 Finding out what keys ssh is using

Finding what keys ssh is offering to the server is fairly simple. Run *ssh -vT cloudnode@git.cloudno.de* and look at the output:

```
debug1: Authentications that can continue: publickey
debug1: Next authentication method: publickey
debug1: Trying private key: /home/user/.ssh/id_rsa
debug1: Trying private key: /home/user/.ssh/id_dsa
debug1: No more authentication methods to try.
Permission denied (publickey).
```

8.6 SSH private key passphrases

Using a private key without a passphrase is basically the same as writing down that random password in a file on your computer. Anyone who gains access to your drive has gained access to every system you use that key with. The solution is obvious, add a passphrase.

```
$ ssh-keygen -p
Enter file in which the key is (/home/user/.ssh/id_rsa):
Key has comment '/home/user/.ssh/id_rsa'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
```

When you use a passphrase protected key, you need to add it to your ssh-agent. It stores it securely and you don't have to reenter your passphrase. The use of the agent is mandatory, as on most systems git eats stdin and you won't be able to type in your passphrase during git operations.

8.7 Analysing the application log files

Whenever your application throws an exception, it will be logged to you applications's main log file. You can also use the **console.log()** instruction to write to that file.

You can view the log file from the "My Apps" management page. You can also use the command line to tail your application log file. Run the "cloudnode app logs" command from your application directory.

```
$ cloudnode app logs

cloudnode info Checking config..
cloudnode info Munging require paths..
cloudnode info Globallizing Buffer
cloudnode info Reading file...
cloudnode info Cloudnode wrapped script starting (32623) at  Wed, 06 Apr 2011 23:17:40 GMT
cloudnode info [INFO] You asked to listen on port 8080 but cloudnode will use port 8007 inst
cloudnode info Server running
cloudnode info
```

8.8 Cloudnode Support Group

For additional help visit the Cloudnode Support Group[3].

8.9 Contact Us

You can also use the contact form from every page's footer to send us a private message.

[3]http://support.cloudno.de

Chapter 9

Node.js

Current version: 0.4.10 (08/07/11) (see also: how to update, changelog)

Node.js is a non-blocking, evented I/O framework using the V8 JavaScript engine.

We host Node.js applications in their own VMs, which we call Node Machines. Cloudnode is based on the Nodester platform. During the beta stage, we may need to shut the service down for upgrades or service without notice. You should join our discussion group[1] to share any feedback and get important announcements that can affect running apps.

- Preparing for deployment (**??**)
- Deploying to Cloudnode (**??**)
- Dependencies (section 9)

Preparing for deployment

You can name the main file of your app like you want. Just make sure to specify that name, when you use the "create app" command. You can also specify any port you like as parameter to the listen function (in this sample port 8124). The platform will transparently override the parameter with the port of your Node VM which is in effect.

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(8124, "127.0.0.1");
console.log('Server running at http://127.0.0.1:8124/');
```

Whenever a port needs to be specified as a parameter for other function calls, use the environment variable "app_port" to use the port in use by your Node VM like in the following example:

[1] http://groups.google.com/group/cloudnode

```
var webrepl = require('webrepl');
var port = process.env["app_port"];

webrepl.start(port);
console.log("Web repl started (Listening on port " + port + ")");
```

Deploying to Cloudnode

To create an app with the the Cloudnode command line use the following command:

```
$ cloudnode app create <appname> <startfile>
          - Creates a new app named <appname>, <startfile> is optional
```

Deployment is done with every Git push command. Additionally the application is restarted to active the changes introduced with the push command.

Dependencies

When your repository includes sub modules, these are also push to the Node Machine.

Chapter 10

CouchDB

The Cloudnode platform offers a CouchDB database option to store persistent data. The database is connected through a fast local connection. We also provide a SSL secured remote connection, which can be used to access the database remotely or for replication.

CouchDB Administration

The "My Databases" tab lists your databases. From here you can also create new databases, delete existing database or manage your databases using the Futon Web GUI. You can add sharing rules and manage your map reduce functions. Clicking on a database name opens the database details page, where you find API keys and URLs to externally access you database.

Usage Example

We plan to provide example apps that demonstrate how to use Node.js and CouchDB. We are currently preferring cradle to access CouchDB. Cradle can be installed using npm. For details on using npm with your Node VM see Node package manager[1].

[1] /node-package-manger

Chapter 11

SSL

Secure Sockets Layer is a protocol to encrypt traffic over the internet. We support SSL to access the API[1]. To use SSL use the following URL:

> https://secure.cloudno.de/

We are working on a solution to allow every hosted application to use SSL.

[1] /api

Chapter 12

HTTP Caching

All hosted NodeJS apps leverage a HTTP cache on Cloudnode. We are using Varnish[1] as a HTTP accelerator. Varnish uses numerous techniques to improve the performance. It even has compiled configuration libraries, in-memory logging, and works hand in hand with the OS for memory management.

Cache Invalidation

The biggest challenge when using caching is the control of the cache TTL and cache invalidation. Varnish only delivers a cached page, when it is absolutely safe to do so. The URL and all parameters have to match for a cache hit. So a Node app can control caching in different ways. Per default no caching takes place, but you should decide which pages or parts of your app could benefit from caching. Your application's response time will improve dramatically when you take the advantage of caching.

Cache TTL Control

Every application that is hosted on Cloudnode, even when running on a custom domain, leverages the caching layer. The time to live, or TTL, can easily be controlled using Node's response.setCacheable() call. By default, pages are not cached. When set to true, pages are cached for a long time, a perfect choice for pictures and other static content. When set to undefined, the cache headers need to be supplied by the application. This option allows custom control of the cache TTL times.

[1]http://www.varnish-cache.org/

Chapter 13

Cloud Storage

Every Node application on Cloudnode runs in a virtual machine[1] with its own disk storage. The disk holds a full checkout of the application's Git[2] repository. So you have access to modules sub-modules and static resources, you have checked in. The disk also holds all Node modules you have installed using the npm[3] command and a minimum set of directories to get Linux running like /**etc**.

Special Directories

You should not assume, that there is write access to the disk except to the /**mnt** and /**tmp** directories. Your application can use these directories to store dynamic data. As the names suggest, /**tmp** can be used to store volatile files and /**mnt** can be used to store permanent files. These will always "follow" your application and be mounted at /**mnt**.

[1] node-vm
[2] /git
[3] /node-package-manager

Chapter 14

Custom Domains

Cloudnode allows you to have Node apps written and hosted on Cloudnode respond to requests from your own domain names. For example, if you own the domain hardtoremember.com and the Node app hardtoremember.cloudno.de, then you can configure things such that visitors to hardtoremember.com are served the Node app. The app is still created and hosted on Cloudnode servers.

To accomplish this, you need to:

- Own a domain.
- Configure it to point to the Cloudnode platform.

To own a domain, you can use any popular registrar service such as GoDaddy or EasyDNS. To configure it to point to the Cloudnode platform, you need to create a "CNAME" record. See instructions for how to do this with GoDaddy or with EasyDNS.

DNS can be tricky, so if you need help, just ask in the Cloudnode group.

Example CNAME Record

    www.example.com. CNAME example.cloudno.de

Make sure to use the symbolic name and not an IP address, because IP addresses are changing.

Steps to activate a custom domain on Cloudnode

Custom domains are added as routes to you application using the Cloudnode command line[1]. To add the domain www.example.com to your application:

1. Go to your configured application directory
2. Run the following command line:

[1] /cloudnode-command-line

```
    $ cloudnode appdomain add www.example.com
```

After these steps your application should be reachable under the URL http://www.example.com.

Chapter 15

Deploy Hooks

Deploy hooks are used internally on Cloudnode to checkout the latest version into your Node VM. They are currently not supported for custom actions.