

1. Documentation	2
1.1 Getting started	2
1.1.1 Amazon EC2	7
1.1.2 Cloud Foundry	11
1.1.3 Heroku	13
1.1.4 OpenShift	14
1.1.5 Rackspace	15
1.1.6 Cloud9 IDE	15
1.1.7 Digital Ocean	16
1.1.8 Elastic Beanstalk	17
1.1.9 Nodejitsu	18
1.2 What's new	20
1.2.1 Updating to the latest version	20
1.3 Frequently asked questions	21
1.4 Glossary	24

Documentation

StrongLoop Suite documentation

StrongLoop Suite includes:

- [LoopBack](#), an open-source mobile backend framework for Node.js.
- [StrongNode](#), professional support for Node.js, plus cluster management and other powerful modules.
- [StrongOps](#), a built-in monitoring and management console.

To get up and running quickly, see [Getting started](#).

What's new

LoopBack now enables you to send push notifications to mobile apps. See [Creating push notifications](#) for details. Client SDKs have been updated to support push notifications:

- [Android SDK](#) (version 1.2)
- [iOS SDK](#) (version 1.2)

StrongLoop now supports the [Digital Ocean](#) cloud platform.

See [What's new](#) for a complete list.

Getting started

▼ [Uninstall StrongLoop Suite if you previously installed a binary....](#)

If you previously downloaded and installed StrongLoop Suite, uninstall it as follows:

- **Windows:** Use **Control Panel > Programs > Programs and Features**.
- **OS X:** Run the uninstall script that was installed with StrongLoop Suite: `/usr/local/bin/uninstall-strongloop-suite.sh`.
- **Linux:** Use the standard method for your flavor of Linux; for example, on Red Hat, `rpm`.



Note for OS X Users

If you didn't install StrongLoop Suite, but have `strongloop-node` version 1.1.4-1 or earlier, you must [download the OS X uninstall script](#).

▼ [Install Node.js....](#)



If you already installed Node, you can skip this step.

Windows and OSX: [Download the native installers from nodejs.org](#).

Linux: Use the appropriate native package:

For RPM-based distros, use `yum`:

```
$ sudo yum install nodejs npm
```

For Ubuntu-based distros, use `apt-get`:

```
$ sudo add-apt-repository ppa:chris-lea/node.js
$ sudo apt-get update
$ sudo apt-get install nodejs
```

▼ Install StrongLoop tools...

Install `slc`, the StrongLoop command line tool, with the following command:

```
$ npm install -g strong-cli
```

On some systems, you may need to run the command with system privileges:

```
$ sudo npm install -g strong-cli
```

This tool enables you to quickly create and scaffold LoopBack applications, and provides other capabilities. Follow the instructions in the next two sections to create and run the StrongLoop example and create your own LoopBack application.

For complete reference documentation, see [Command-line reference \(slc lb\)](#).

▼ Create and run the LoopBack sample app....

LoopBack is an open-source backend framework built on Node.js that enables you to connect mobile applications to your data. For an overview of LoopBack, see [StrongLoop | LoopBack](#).

The LoopBack sample application is a server-side only application that provides an introduction to LoopBack application development. If you prefer, you can create your own LoopBack app from scratch; see [Creating your own LoopBack application](#).



You must have the `git` command-line tool installed to create the sample application. If needed, download it at <http://git-scm.com/downloads> and install it.

On Linux systems, you must have root privileges to write to `/usr/share`.

Enter the following command to create the Loopback sample application:

```
$ slc example
```

This command creates the sample application in a new directory named `sls-sample-app` and installs all of its dependencies.

Run the sample application

Enter the following command to run the sample application:

```
$ cd sls-sample-app
$ sl-run
```

View the sample application in a browser at <http://localhost:3000>. You can also view the generated API at <http://localhost:3000/explorer>.

For more information on the LoopBack sample application, see [Using the LoopBack sample app](#).

After you've had a look, stop the application by pressing **Ctrl-C**.

▼ Create your own LoopBack application...

Now that you've experimented with the LoopBack sample app, you can start to build your own app.

Follow the steps in this section to create a minimal LoopBack application that provides a skeleton on which you can build your own application.

Create a new application

Enter this command to create a new blank template LoopBack application:

```
$ slc lb project myapp
```

Replace "myapp" with the name of your application. This command creates a new directory called `myapp` (or the application name you used) that contains all the files and directories to provide "scaffolding" for a LoopBack application.

Run empty application

At this point, you have an "empty" LoopBack application. It won't actually do much, but you can run it to get a sense of what LoopBack scaffolding provides. Enter these commands:

```
$ cd myapp
$ slc run app
```

Of course, substitute your application name for "myapp."

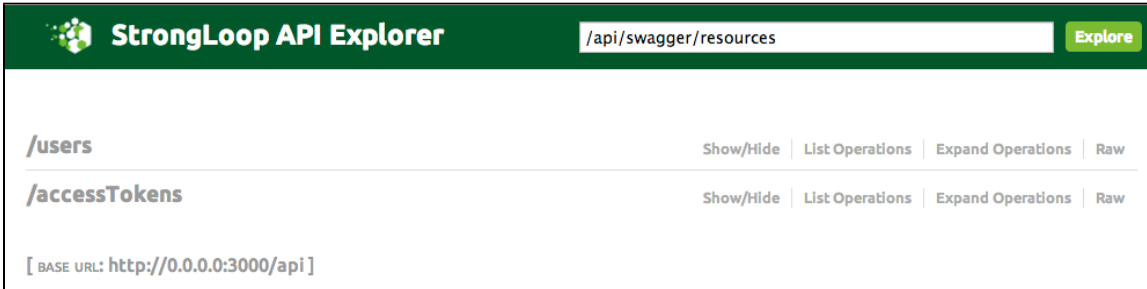
Now, load this URL in your browser: <http://0.0.0.0:3000/>

You'll see the basic JSON data returned by the empty app; something like this:

```
{
  "started": "2014-01-04T00:00:58.383Z",
  "uptime": 15.991
}
```

Load this URL in your browser to view the API explorer: <http://0.0.0.0:3000/explorer>.

The API explorer will show you the REST APIs of the built-in user and access token models.



A quick look under the hood

As mentioned before, `slc` creates a number of files and directories for a LoopBack application; specifically:

- `app.js` - application main program file.
- `app.json` - configuration file that specifies the TCP port and IP address where the application listens.
- `datasources.json` - data source configuration file.
- `/models` directory - The application will load all JavaScript files in this directory when it starts. See [Initializing the application](#) for details.
- `models.json` - Contains model definitions created with `slc lb model`. You can also edit this file manually to add models. See [Creating models](#) for more information.
- `/node_modules` directory, containing a directory for each Node module upon which LoopBack depends by default. See [npm-folders](#) for more information.
- `package.json` - application metadata file that identifies the project, lists project dependencies, and provides other metadata such as a description, version and so on. For more information, see [package.json](#).

Now you can add models and other code to create your LoopBack application. If you want to explore LoopBack further, see [Creating a LoopBack application](#).

▼ Use StrongOps to monitor LoopBack sample app performance....

StrongOps is a performance monitoring and devops solutions for Node.js apps. For an overview of StrongOps, see [StrongLoop | StrongOps](#).

To enable StrongOps application monitoring for the LoopBack sample application:

1. If you have not already done so, create an account on [strongloop.com](#) and login.
2. In the root directory of your app, enter this command to register a new StrongOps account:

```
$ cd sls-sample-app
$ slc strongops --register
# If you already have a StrongOps account, don't use --register.
```

You'll be prompted to register with StrongOps if you haven't already done so. Enter the same email address and password you used when you registered on [strongloop.com](#).



You *must* use the same email and password when you create accounts for StrongOps and strongloop.com. (we're working to fix this soon!) If you previously registered a StrongOps account with the same credentials as your strongloop.com account, omit `--register` and enter those credentials when prompted.

Run the sample app again

Now that you've enabled StrongOps monitoring, restart the sample app. As before, enter this command in the `sls-sample-app` directory:

```
$ sl-run
```



If you don't have a C compiler (Visual C++ on Windows or XCode on OSX) and command-line "make" tools installed, you will see some error messages when you run the app. You will still be able to run the sample app and monitor it with the StrongOps Console, but you won't be able to use some features such as memory profiling.

Simulate application load

To get a sense of the metrics you can view with the StrongOps dashboard, you need to generate some load on the application. The sample application comes with a load generator script to simulate load against the REST API. Enter these commands to run it:

1. Open a new shell window.
2. Run the load generator.

```
$ cd sls-sample-app
$ sl-run bin/create-load.js
```

Check out StrongOps dashboard

Now, log in to the [StrongOps console](#). You'll need to wait a few minutes to see results from the load script against the sample app.

Explore the application metrics that the console provides. For more information on the StrongOps console, see [Using the StrongOps console](#).

▼ Use StrongOps to monitor your existing Node app...

StrongOps is a performance monitoring and devops solutions for Node.js apps. For an overview of StrongOps, see [StrongLoop | StrongOps](#).

To monitor an existing Node app, you'll first need to create a StrongOps account:

1. If you have not already done so, create an account on [strongloop.com](#) and login.
2. In the root directory of your app, enter this command to register a new StrongOps account:

```
$ cd your-app-dir
$ slc strongops --register
# If you already have a StrongOps account, don't use --register.
```

You'll be prompted to register with StrongOps. Enter the **same** email address and password you used when you registered on strongloop.com. After you register, slc will generate your API key and save it to `strongloop.json` in your application's directory.



You *must* use the same email and password when you create accounts for StrongOps and strongloop.com. (we're working to fix this soon!) If you previously registered a StrongOps account with the same credentials as your strongloop.com account, omit `--register` and enter those credentials when prompted.

Now to set up your Node application for StrongOps monitoring, follow these steps:

1. **Add strong-agent as a dependency** in your application's `package.json` file. The easiest way to do this is to use the following command in the application directory:

```
$ npm install --save strong-agent
```

This command adds strong-agent as a dependency to the application's `package.json` file.

2. **Call `profile()` in the application code.** To enable StrongOps monitoring, call the strong-agent `profile()` function at the very beginning of your app, before any other `require()` calls. Add the following call at the beginning of your module:

```
require('strong-agent').profile();
```

Run your application and monitor

Now run your application as you normally do.



If you don't have a C compiler (Visual C++ on Windows or XCode on OSX) and command-line "make" tools installed, you will see some error messages when you run the app. You will still be able to run the sample app and monitor it with the StrongOps Console, but you won't be able to use some features such as memory profiling.

Log in with your username and password at <http://strongloop.com/>. The StrongOps console is at <http://strongloop.com/ops>.

To see any interesting results in the StrongOps console, you'll need to let the application run some time. Also, you may need to generate some load on the application (for example, HTTP requests) to see interesting metrics.

For more information on the StrongOps console, see [Using the StrongOps console](#).

▼ Deploy to cloud platform....

To make your application available via a cloud provider, follow the instructions for your platform:

- [Amazon EC2](#)
- [Cloud Foundry](#)
- [Heroku](#)
- [OpenShift](#)
- [Rackspace](#)
- [Cloud9 IDE](#)
- [Digital Ocean](#)
- [Elastic Beanstalk](#)
- [Nodejitsu](#)

Download mobile client SDKs

Use the LoopBack iOS and Android SDKs to create native mobile apps that use the LoopBack API. See:

- [Download Android SDK](#)
- [Download iOS SDK](#)

- [Client SDK documentation](#)

Amazon EC2

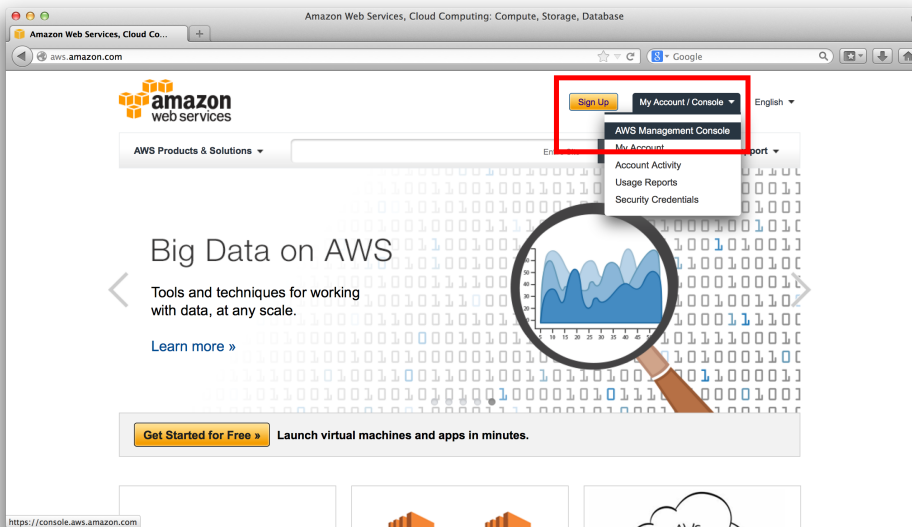
- [Creating an instance](#)
- [Connecting to the instance](#)
- [Verifying the StrongLoop app](#)

You can deploy StrongLoop applications directly as an Amazon Machine Image (AMI), which comes preinstalled with Node and the `slc` command-line tool to simplify the deployment to AWS EC2.

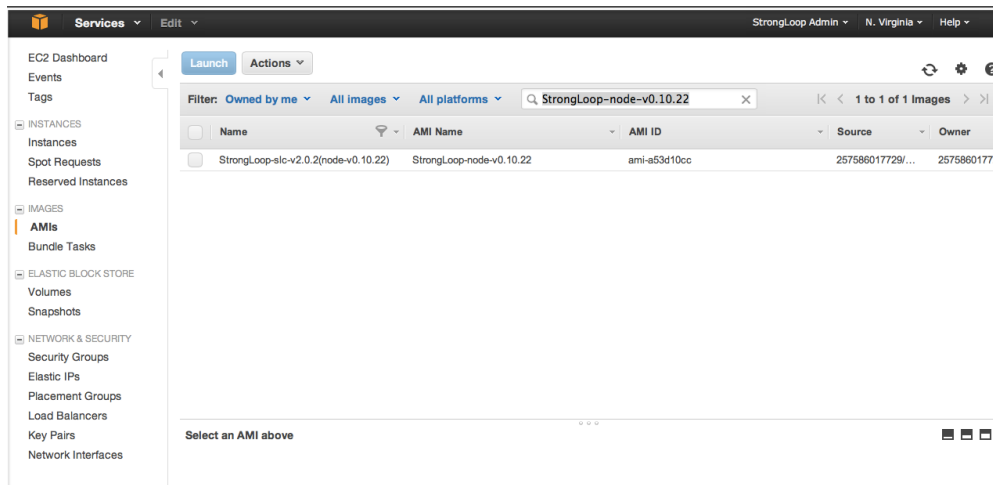
Creating an instance

Follow these steps to create a new AWS EC2 instance with StrongLoop tools installed:

1. Log into your Amazon AWS account at <http://aws.amazon.com/>, and go to **My Account/Console > AWS Management Console**.

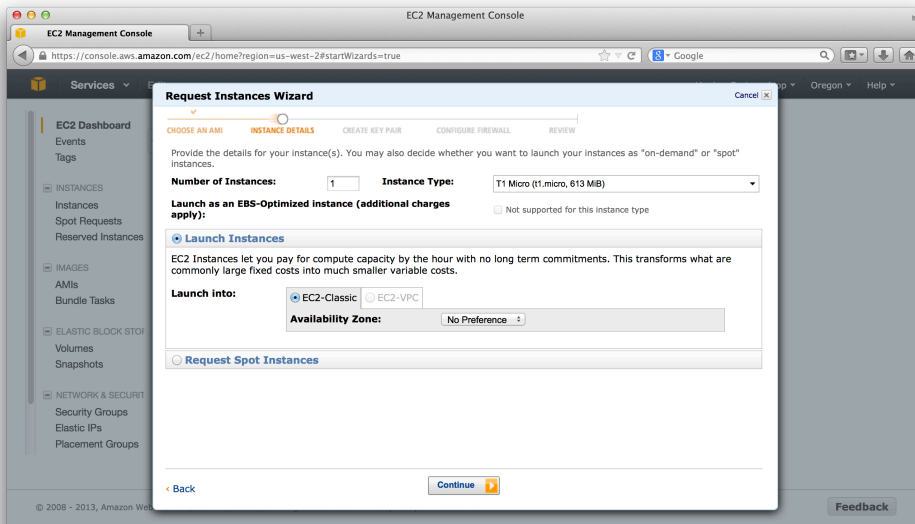


2. Click **My Account/Console > AWS Management Console > EC2 > Images (AMIs)**. Search for StrongLoop-node-v0.10.22 and launch the StrongLoop AMI

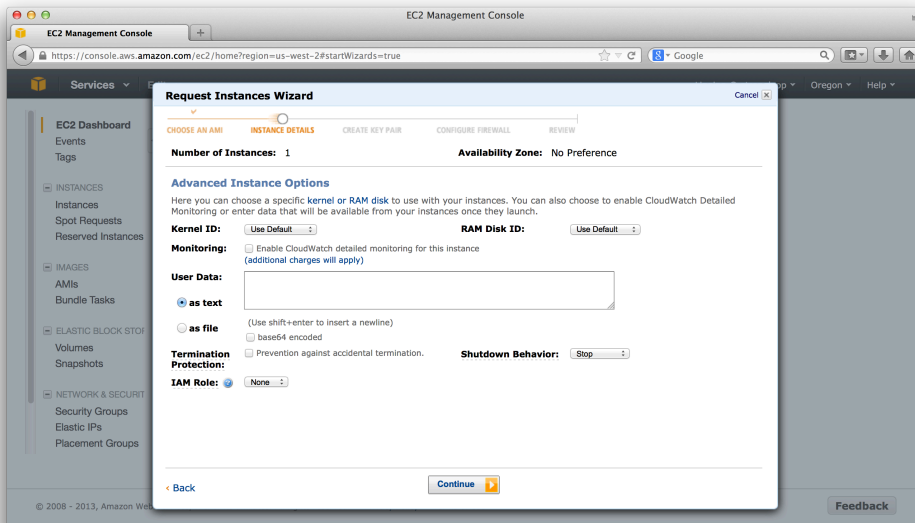


3. Configure instance details. Use default settings to configure your instance as follows

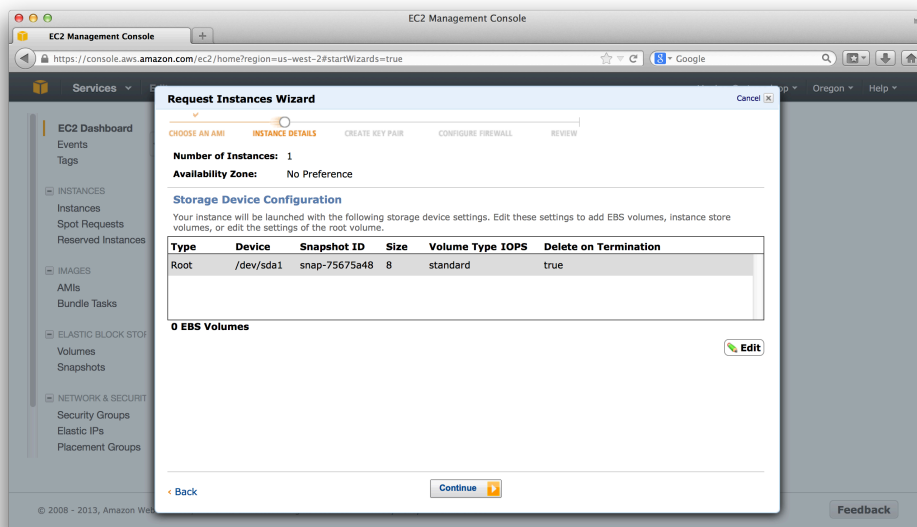
Basic options



Advanced options

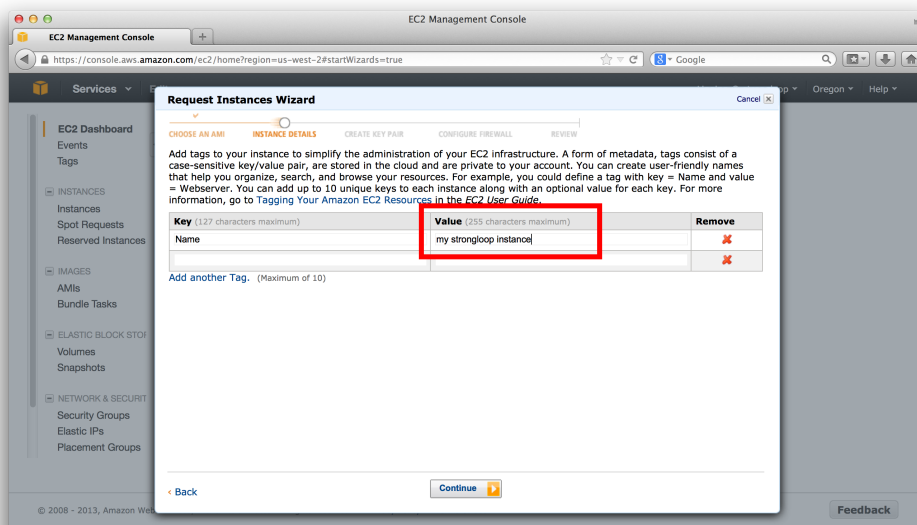


Storage Device



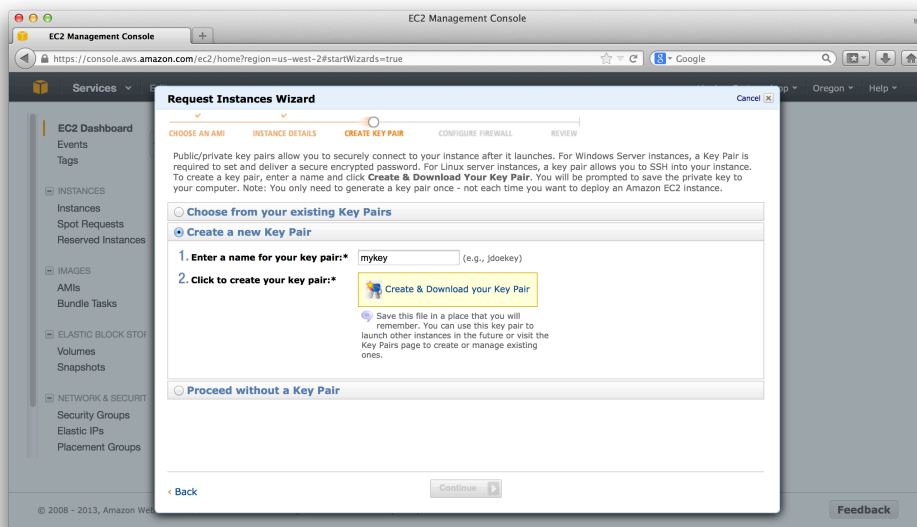
Instance Tags

Tag 'my strongloop instance' as the name of this instance. It will help you to manage all your resources.



4. Create a new key pair to secure the connections.

Key in your key pair name: `mykey` and click **Create & Download your Key Pair** to download the newly-generated key pair to your local drive. If you've previously generated a key pair, you can also choose it instead.



5. Configure security.

Services Edit Qihan Zhang Oregon Help

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group ☐ Select an existing security group

Security group name:

Description:

Protocol	Type	Port Range (Code)	Source
SSH	TCP	22	Anywhere 0.0.0.0/0
Custom TCP Rule	TCP	0	Custom IP (e.g., 192.168.2.0/24, sg-)

Add Rule

Warning

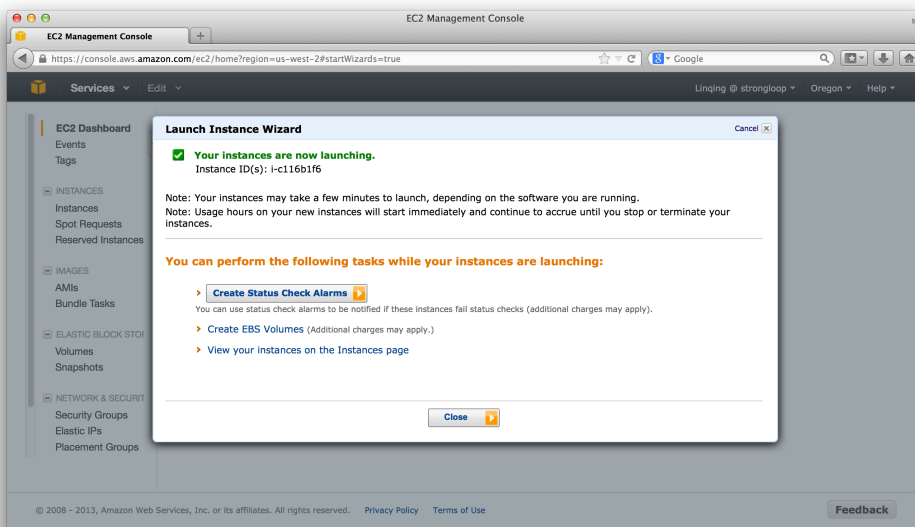
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend creating security group rules to allow access from known IP addresses only.

Cancel Previous **Review and Launch**

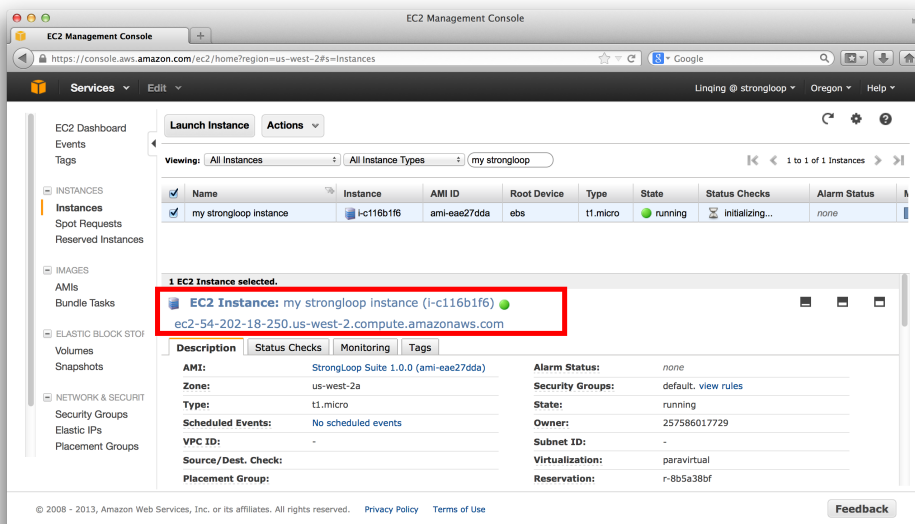
© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#) [Feedback](#)

6. Click **Review and Launch** after you review your instance configuration.

7. It may take several minutes to launch your instance.



8. Check out the new instance you created. Click **EC2 Dashboard > INSTANCES > Instance**, and you will a new instance called "my strongloop instance" has been added to your instance list.



Connecting to the instance

After you have configured the network & security settings, you can use `ssh` to log into the instance. You can find the host name in the instance details (refer to the screenshot above). For example:

```
$ ssh -i /path/to/mykey -l ec2-user ec2-54-202-18-250.us-west-2.compute.amazonaws.com
```

Node version 0.10.22 and `slc` will be available on this instance.

Follow the instructions in [Using the LoopBack sample app](#) to create and run a LoopBack app.

Verifying the StrongLoop app

Once you create and run the sample application, you can verify it's successful deployment via the instance's public DNS IP. For example:

```
ec2-54-202-18-250.us-west-2.compute.amazonaws.com:3000
```

Cloud Foundry

- [Creating an application](#)
- [Create Procfile and commit](#)
- [Push application to Cloud Foundry](#)



Follow the steps in [Getting started](#) to install Node and the StrongLoop command-line tool, then create a StrongLoop application on your local system.

Then follow the steps below to deploy the app to Cloud Foundry.

Creating an application



Cloud Foundry currently supports Node v. 0.10.21, but StrongLoop uses 0.10.22. Change the Node version by adding the following to the app's `package.json` file:

```
{ "engines" : { "node" : "0.10.22" }}
```

Create Procfile and commit

1. Create a `Procfile` in the root directory of your app that contains the following:

```
web: node app.js
```

2. To deploy, add your application to a Git repository.

```
$ git init
$ git add *
$ git commit -a -m "Initial Commit"
```



For faster deployment time, delete the application's `node_modules` directory.

Push application to Cloud Foundry

1. Login to [Cloud Foundry](#). Create an account first if you don't already have one.
2. Install the `cf` command line tool, and read the [Cloud Foundry CLI guide](#).

```
$ gem install cf
```

3. Select the target CloudFoundry instance:

```
$ cf target api.run.pivotal.io
```

4. Login into the Cloud Foundry platform-as-a-service:

```
$ cf login
```

5. When you are satisfied that everything is OK, push your app to Cloud Foundry. Pick a name for your app and run the command:

```
$ cf push <your app name>
```

That's it! You can now check out your application at the app URL/domain you set for your Cloud Foundry app. If you have enabled StrongOps monitoring, log in to [strongloop.com](#) and go to the [StrongOps dashboard](#). You should be able to see your app.



The first time you run `cf push`, you must specify all the parameters to create a new application at Cloud Foundry. Default values will work fine if you just keep pressing **Enter**. Subsequently, you can just use the `cf push` command; you don't need to specify all the options.

For more information on using Cloud Foundry, see the [StrongLoop blog post on using CloudFoundry](#).

Heroku

- [Create Procfile and commit](#)
- [Push to Heroku](#)
- [Check your dashboard on Heroku](#)



Follow the steps in [Getting started](#) to install Node and the StrongLoop command-line tool, then create a StrongLoop application on your local system.

Then follow the steps below to deploy the app to Heroku.

Create Procfile and commit

1. Create a `Procfile` in the root directory of your app that contains the following:

```
web: node app.js
```

2. To deploy, add your application to a Git repository.

```
$ git init
$ git add *
$ git commit -a -m "Initial Commit"
```



For faster deployment time, delete the application's `node_modules` directory.

Push to Heroku

The StrongLoop Heroku add-on provisions a monitoring account on [StrongOps](#)

1. If you haven't already done so, register an account on [Heroku](#).
2. Install the [Heroku Toolbelt](#).
3. Login with the Heroku command line:

```
$ heroku login
```

4. Once you are ready to deploy your app, follow these steps:

- a. Create your Heroku app with the first command below. When it completes, push to Heroku master to complete the installation of Node on your dyno.

```
$ heroku apps:create <optional_name>
$ git push heroku master
```

- b. Test it with this command:

```
$ heroku open
```

Check your dashboard on Heroku

Once you have created your app, it's time to look at the instrumentation.

1. Go to [Heroku](#) and find your app.
2. Click **Heroku app dashboard** to view the various dynos and add-ons for your app.
3. Click **StrongLoop add-on** to view the StrongOps Control Panel.
4. Click **StrongOps Dashboard** to access the StrongLoop Ops dashboard.

OpenShift

The StrongLoop OpenShift Cartridge provides Node and the StrongLoop command-line tool (slc).

Trying the StrongLoop sample application

Follow these steps to get started using StrongLoop on OpenShift:

1. Login at <http://www.openshift.com/>. Create an account if you don't already have one.
2. Ensure you have the latest version of the [client tools](#). As part of this process, you'll run the `rhc` setup command and choose a unique name (called a *namespace*) that becomes part of your public application URL.
3. Create an application on OpenShift with the following command:

```
rhc create-app yourapp
https://raw.github.com/strongloop/openshift-cartridge-strongloop/master/metadata/
manifest.yml
```

Replace "yourapp" with your application name. You'll see the message:

```
Creating application 'yourapp' ...
```

The first build will take a few minutes, so please be patient.

When it's done, you'll get a folder called `yourapp` (or whatever you chose for your app name) in your current path containing the LoopBack sample app. You'll see a friendly welcome:

```
Your application 'yourapp' is now available.

URL:          http://yourapp-<namespace>.rhcloud.com/
SSH to:       527...0069@yourapp-<namespace>.rhcloud.com
Git remote:   ssh://527...0069@yourapp-<namespace>.rhcloud.com/~/.git/yourapp.git/
Cloned to:    <local-path>/yourapp
```

Where:

- `<namespace>` is your OpenShift namespace.
- `<local-path>` is the path to your app's source git clone on your local system.

You can now try out the LoopBack sample application running on OpenShift:

- View the LoopBack sample app at the indicated URL that looks like `http://yourapp-<namespace>.rhcloud.com/`.
- View the LoopBack API explorer at `http://yourapp-<namespace>.rhcloud.com/explorer`.

Deploying your own application to OpenShift

If you created your own LoopBack application, follow these steps to deploy the app on OpenShift.

1. Get the remote git url of the OpenShift app by running

```
$rhc app show osapp | grep Git
```

This will return something like this:

```
Git URL:
ssh://52a17af15004464d950003bd@openshiftapp-domain.rhcloud.com/~/.git/openshiftapp
.git/
```

2. In your LoopBack app directory, enter this command, where <remote_git_url> is the URL returned above:

```
$ git remote add openshift <remote_git_url>
```

3. Deploy as follows:

```
$ git push --force openshift master
```

That's it! You can now check out your application at the app URL/domain you set for your OpenShift app.

If you have enabled StrongOps monitoring, log in to strongloop.com and go to the [StrongOps dashboard](#). You should be able to see your app.

Rackspace

To configure and deploy StrongLoop Suite from the Rackspace Control Panel:

1. Create an account at [Rackspace](#).
2. Log into your [Rackspace Control Panel](#).
3. Select the **Deployments** Tab and start 'Create Deployment' Step 1 of 2.
4. Assign a deployment name and region.
5. Select **StrongLoop** blueprint.
6. Configure your StrongLoop options:
 - site address
 - system username
 - system password
 - application name
7. Select **Create Deployment**.
8. Verify installation by opening a browser to your machine IP address or configured domain address.

For more information, see [Rackspace Deployment Service](#) .

Cloud9 IDE

The Cloud9 IDE workspace is your virtual development environment in the cloud.

Prerequisite

If you haven't already done so, [create a StrongLoop account](#), then make sure you're logged in to the StrongLoop website.

Procedure

Follow these steps:

1. [Launch your Cloud9 workspace](#).
2. Click on the green Run button in the Cloud9 IDE toolbar. The StrongLoop Suite sample app will begin running, and the Output tab at the bottom of the window will show:

```
Your code is running at 'http://<your_workspace_name>-c9-<userid>.c9.io'.
....
```

This indicates the application has run successfully. If that's not the case; for example, if you see an error message, then contact us (support@strongloop.com) and we'll gladly give you a hand.

You will also see the sample app web page displayed in the browser pane on the right side of the Cloud9 IDE. If you don't see the browser pane,

click the blue link in the message shown in the Output tab.

Next steps

Now that you have the sample app running:

- For more about Cloud9 and the sample app, see the README file displayed in the Cloud9 IDE.
- Click the big green **GET** button in the browser pane. You'll see the JSON that the app returns from various GET requests to the app's REST API.
- Scroll down in the browser pane. You'll see suggestions for further next steps.

Digital Ocean

Follow these steps to install a 512 MB RAM, Ubuntu 12.04 droplet with Node and the StrongLoop command-line tool:

1. [Sign up](#) for Digital Ocean if you haven't already.
2. Install the latest version of Node.js. Log into your Droplet and enter the following commands.



There are several ways to install Node; The following method is simple and will get you up and running quickly. For alternative methods, see [How To Install an Upstream Version of Node.js on Ubuntu 12.04](#).

```
$ sudo apt-get update
$ sudo apt-get install python-software-properties python g++ make
$ sudo add-apt-repository ppa:chris-lea/node.js
$ sudo apt-get update
$ sudo apt-get install nodejs
```

3. Enter this command to verify your Node installation:

```
$ node -v
```

You should see a result like

```
v0.10.24
```

If you see anything else (such as an error message), then there was a problem installing Node. Please [contact Digital Ocean](#) for help.

4. Install the StrongLoop command-line tool (slc) by entering the following command. If you get permission errors, re-run the command with sudo.

```
$ npm install -g strong-cli
```

5. Enter the following command to verify your installation:

```
$ slc version
```

You should see a result like

```
slc v2.1.0 (node v0.10.22)
```

If you see anything else (such as an error message), then [contact StrongLoop support](#) for help.

6. Install the sample LoopBack application:

```
$ slc example
$ cd sls-sample-app
```


7. Install and setup the StrongOps agent:

```
$ slc strongops --register
```

Provide your full name, email and password when prompted.

8. Create a StrongLoop account to access your StrongOps dashboard.

Go to <https://strongloop.com/register/>

Provide your name, email address, and password.



You *must* provide the same email address and password that you used in Step 7.

9. View your StrongOps dashboard at <http://strongloop.com/ops/dashboard>.



You will not see any data until your application has been running for a few minutes. So, please make sure to execute the next step.

10. Run the sample application:

```
$ slc run .
```

View the LoopBack sample application by visiting the external-facing IP that Digital Ocean assigned to your droplet at port 3000:

`http://[Droplet-IP]:3000`

Elastic Beanstalk

Prerequisites

Follow the instructions in [Getting started](#) to install Node and the `slc` command-line tool on your local system and use `slc lb` to create a LoopBack application locally. It's usually a good idea to test the application on your local system also.

If you wish, follow the instructions in [Getting started](#) to enable StrongOps performance monitoring for your application.



Be sure you have the latest version of the `slc` command-line tool on your local system. If you've already installed it, enter this command to update to the latest version:

```
npm update g strong-cli
```

Running your application

Follow these steps to run an application on [Elastic Beanstalk](#):

1. Compress your application into a zip file.



In the zip file you create, include all the files and sub-directories in your application directory except `node_modules`. This directory and its contents are large and not required. Also make sure that `package.json` and the main JavaScript file `app.js` are in the root level of the zip file.

2. Open the [Elastic Beanstalk Management Console](#).
3. In the upper right of the Elastic Beanstalk console, click **Create New Environment**
 - a. Enter a name and description for your application.
 - b. For **Environment tier** choose **Web Server**.
 - c. For **Predefined configuration** choose **Node.js**.
 - d. For **Environment type** choose **single instance** and click **Continue**.
 - e. Under **Application Version**, Choose **Upload your own**, click **Choose File**, then select the zip file you created previously.
 - f. Enter a name, URL and description for your environment and click **Continue**.
 - g. Select **Additional Resources** if you need them then click **Continue**.
 - h. Under **Configuration details**, select **instance type** as **micro**. If you need SSH access to your virtual machine, select a key pair or create one. Enter all the other information such as email address you require then click **Continue**.
 - i. Review all the information then click **Create**.

4. This will start your LoopBack application.



By default, Elastic Beanstalk uses an older version of Node.js that is not compatible with StrongOps. Therefore, you must upgrade to the latest version. For the latest version of Node supported by Elastic Beanstalk, see [Supported Platforms - AWS Elastic Beanstalk](#).

5. Change the version of Node in your environment by clicking **Configuration > Software Configuration** and edit the Node version to latest version.
6. You can access your app by clicking on the link `<environment_name>.elasticbeanstalk.com/explorer` shown in the title of the management console.
7. If you enabled StrongOps monitoring, you can then view application metrics in the the [StrongOps dashboard](#) once you're logged in.

Nodejitsu



Follow the steps in [Getting started](#) to install Node and the StrongLoop command-line tool, then create a StrongLoop application on your local system. And:

- Register with StrongOps and set up the sample app for monitoring.
- Run the sample application.

Then follow the steps below to deploy the app to Nodejitsu.

To deploy the application on Nodejitsu:

1. Sign up for Nodejitsu at <https://www.nodejitsu.com/signup/>
2. Install the jitsu command line tool (CLI).

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

3. Execute the user confirmation command and code you got in your signup email. It should look something like:

```
$ jitsu users confirm [user name] [a bunch of numbers and letters]
```

4. Login. You'll be prompted for your password to login.
5. Deploy the StrongLoop sample application to Nodejitsu. Inside of the sls-sample-app directory execute:

```
$ jitsu deploy
```

Enter a subdomain name when prompted.

6. Next, you'll get prompted for a Node engine. Choose one and confirm your choice.

At this point Nodejitsu will do a few checks, upload your app and then give you a URL and port assignment where you can view it.

StrongLoop Suite Sample Application

Welcome

Sample Requests

GET /api/cars

GET /api/cars/2

GET /api/cars?filter

GET /api/locations

GET /api/locations/nearby

GET /api/locations/id/inventory

Next Steps

API Explorer

Choose Your Own Adventure

StrongLoop Suite

Welcome to StrongLoop Suite, an API tier for connecting Enterprise data to devices and browsers. StrongLoop Suite includes an open source mobile framework, a built-in monitoring and management console, all built on top of a certified distribution of Node.js.

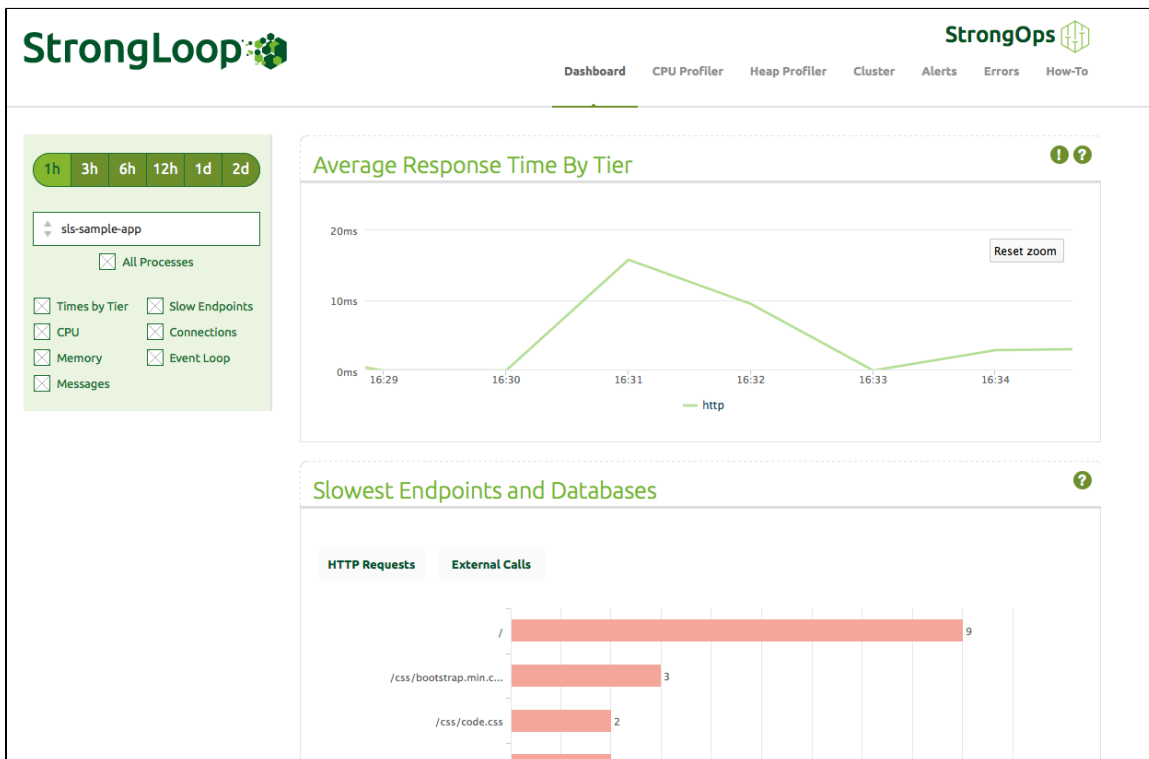
Sample Requests

Click on the friendly GET buttons below to try out a few example requests!

GET /api/cars

You should note that locally, the LoopBack sample application is configured by default on port 3000, while Nodejitsu deploys the app by default on 80.

You should also be able to go back to your StrongOps dashboard and view your app on Nodejitsu being monitored.



If you run into any issues, restart the application. First, enter this command to see the status of your apps:

```
$ jitsu list
```

Then restart the one you want by executing:

```
$ jitsu apps restart [name of app]
```

To stop your app, execute:

```
$ jitsu apps stop [name of app]
```

Redeploy an app (executed from with your apps folder) with:

```
$ jitsu apps deploy
```

And finally, if you've made total scrambled eggs of things, you can destroy an app with:

```
$ jitsu apps destroy [name of app]
```

What's new

Here are the latest additions and improvements to the StrongLoop application framework.

Jan 2014

LoopBack now enables you to send push notifications to mobile apps. See [Creating push notifications](#) for details. Client SDKs have been updated to support push notifications:

- [Android SDK](#) (version 1.2)
- [iOS SDK](#) (version 1.2)

StrongLoop now supports the [Digital Ocean](#) cloud platform.

Dec 2013

StrongLoop Suite is now distributed through npm instead of a downloadable installer or archive file. See [Getting started](#) for details and the [blog post](#) for further context.

New features:

- [LoopBack authorization and authentication](#)
 - [Controlling data access](#)
 - [Access control models](#)
- [iOS SDK](#)
- Support for the following cloud platforms:
 - [OpenShift](#)
 - [Cloud9 IDE](#)
- StrongOps:
 - [Error reporting](#)
 - [Configuring alerts](#)
 - [Controlling an application cluster](#)
 - [Open data API](#)

Updating to the latest version

Generally, npm will automatically update packages that your application requires, based on the information in the `package.json` file. For more information on `package.json`, see the [npm documentation](#).

To get the latest features of the `slc` command-line tool, you need to update it to the latest release.



 The current version of slc is 2.2.2.

What version of slc do you have?

To see the version of `slc` you currently have installed, enter this command:

```
$ slc version
```

How to update

To update to the latest version of `slc`, use the following command:

```
$ npm update -g strong-cli
```

On some systems, you may need to use superuser privileges:

```
$ sudo npm update -g strong-cli
```

Handling peer dependency errors

Sometimes, you may encounter the following error:

```
npm ERR! peerinvalid The package xxx does not satisfy its siblings' peerDependencies requirements!
```

where `xxx` is a package name such as `strong-cluster-control`.

To correct this, simply enter the command:

```
$ npm update -g xxx
```

Where `xxx` is the module in the error message.

Frequently asked questions

- [General questions](#)
 - [What platforms does StrongLoop Suite support?](#)
 - [What is slc?](#)
 - [How do I install the latest version of slc?](#)
 - [What is a "private registry?"](#)
 - [What message queues does StrongLoop Suite support?](#)
- [LoopBack](#)
 - [What are the native SDKs in LoopBack ?](#)
 - [Which data connectors does LoopBack have?](#)
- [StrongNode](#)
 - [Which version of Node.js is StrongNode built with?](#)
 - [What are the modules in StrongNode?](#)
 - [Cluster modules](#)
 - [Other modules](#)
 - [Community modules](#)
- [StrongOps](#)
 - [What endpoints does StrongOps monitor?](#)
 - [Is it possible to get a heap dump while I notice an issue ?](#)
 - [What is Open Data API ?](#)
 - [What is CPU profiling ?](#)

- [Can I do memory profiling with StrongOps ?](#)
- [What is application tracing? Can I monitor my transactions?](#)

General questions

What platforms does StrongLoop Suite support?

StrongLoop Suite is free to use on up to one process in production and an unlimited number in pre-production. It is supported on the following operating system platforms:

- RHEL/CentOS 6.3 (RPM)
- Debian/Ubuntu 12.10 (DEB)
- Mac OS X Mountain Lion 10.8 (PKG)
- Microsoft Windows 8, 2008 (MSI)

Cloud platforms:

- Amazon EC2
- Heroku
- CloudFoundry
- Digital Ocean
- Red Hat's OpenShift
- Rackspace Cloud
- Cloud9

What is slc?

slc is a command line tool for building and managing applications. Features include:

- Support for [private npm repositories](#)
- [Advanced debugging](#)
- [Enabling and managing clusters](#)
- Rapid creation of boilerplate for modules
- Easy generation of web application templates
- Including code to connect to MongoDB backends
- RESTful API code generation
- Boilerplate for command-line applications
- A wrapper for all npm commands
- Ability to run Node scripts

How do I install the latest version of slc?

See [Getting started](#) for instructions on getting the latest version of the slc command-line tool.

See [Updating to the latest version](#) to update to the latest version.

What is a "private registry?"

Node applications use *Node Package Manager* (npm) to automate the process of installing, upgrading, configuring, and removing Node packages. It automatically handles dependencies, so when you install a package, it will automatically install other packages that it requires. The standard public registry is npmjs.org, which maintains a database of node packages (over 47,000 and growing).

A private registry is a "non-public" source and configuration management system for node projects that enable an enterprise to use only the modules required by their project while being able to share the common registry across multiple teams within their organization. These registries allow node projects to whitelist and manage proliferation of module functionalities, versions, and source.

What message queues does StrongLoop Suite support?

StrongLoop Suite uses [strong-mq](#) to provide an abstraction layer over common message distribution patterns and message queue implementations, including rabbitMQ, activeMQ, and cluster-native messaging.

It enables you to write an application using a single message queue API, and then deploy it singly or as a cluster, with deploy-time configuration of the messaging provider. Providers include native node clustering, allowing no-dependency deployment during test and development. Support for other providers is on-going, and you can add pluggable support for new message queue platforms.

LoopBack

What are the native SDKs in LoopBack ?

There are two mobile SDKs for accessing the REST API services generated by the LoopBack framework:

- iOS SDK (Objective C) for iPhone and iPad apps. See [iOS SDK](#) for more information.
- Android SDK (Java) for Android apps. See [Android SDK](#) for more information.

Which data connectors does LoopBack have?

LoopBack provides the following connectors to access enterprise and other backend data systems:

- Oracle
- MySQL
- Mongo DB
- REST
- In-memory

StrongNode

Which version of Node.js is StrongNode built with?

StrongNode supports version 0.10.22 of Node.js.

What are the modules in StrongNode?

StrongNode modules fall into two categories: cluster-related and other modules. Additionally, StrongLoop Suite supports a set of community ("userland") modules.

Cluster modules

The following modules enable Node applications to create child processes that all share the same network port. This enables applications to take advantage of multi-core systems:

- [Strong-cluster-control](#) - Allows for run-time management of cluster processes.
- [Strong-store-cluster](#) - A key-value store accessible to all nodes in a node.js
- [Strong-cluster-connect-store](#) - Provides sessions for [Connect](#) and [Express](#) applications. Manages workers, ensuring the correct number of workers are available; enables you to change the worker pool size without restarting the application.
- [Strong-cluster-socket.io-store](#) - Provides an easy solution for running a socket.io server when using node cluster.
- [Strong-cluster-tls-store](#) - An implementation of TLS session store using node's native cluster messaging. Additionally, provides several different message queue implementations, including cluster-native messaging.
- [Strong-mq](#) - An abstraction layer over common message distribution patterns.

Other modules

- [Strong-cli](#) - The StrongLoop command-line tool.
- [Node Inspector](#) – Debugging interface for Node.js.
- [Strong-task-emitter](#) - Performs an arbitrary number of tasks recursively and in parallel and, using an event emitter, passes the results in an asynchronous message.
- [Strong-remoting](#) - Makes objects and data in your Node application need to be reachable from other Node processes, browsers, and mobile clients.

REVIEW COMMENT

Removed - but should we include it here?

- [Strong-agent](#) - Enables performance monitoring of your node.js application. application services. Including system usage at every moment in time to uncover and resolve issues within the application as they arise.

Community modules

StrongLoop supports a set of commonly-used community modules that provide crucial functionality:

- [Express](#) – Web application framework.
- [Connect](#) – Rich middleware framework.
- [Passport](#) – Simple, unobtrusive authentication.

- [Mongoose](#) – Elegant mongodb object modeling.
- [Async](#) – Higher-order functions and common patterns for asynchronous code.
- [Q](#) – Tool for making and composing asynchronous promises in JavaScript.
- [Request](#) – Simplified HTTP request client.
- [Socket.IO](#) – Cross-browser WebSocket for realtime apps.
- [Engine.IO](#) – Transport layer for real time data exchange.
- [Reggie](#) – Lightweight alternative to a full blown npm registry.

StrongOps

What endpoints does StrongOps monitor?

The endpoints that StrongOps monitors include:

- HTTP Requests – URLs and parameters
- Key value store – Redis
- Memory Object caching – Memcached
- Web-services – SOAP, REST, HTTP
- Relational databases - MySQL, Oracle
- Including code to connect to MongoDB backends
- NoSQL data stores like MongoDB

Is it possible to get a heap dump while I notice an issue ?

StrongOps provides visibility into to total allocated memory heap, the memory currently in use by the application and the frequency of garbage collection. For more information, see [Using the metrics dashboard \(Heap use\)](#).

What is Open Data API ?

You can use the open data API to query performance data from StrongOps for use with proprietary reporting tools and integration into broader operations systems.

For more information, see [Open data API](#).

What is CPU profiling ?

CPU profiling is a feature of StrongOps that collects the CPU footprint of the application, modules, functions and line of code. A profiling session for a host and process can be kicked off from the StrongOps console and we recommend running it for about five seconds to get a comprehensive snapshot of the CPU footprint under production load.

Once the profile is collected, StrongOps provides the following analytics:

- Time distribution of code calls in perspective of CPU footprint over the collection period
- Call chains with insight into multiple child elements
- Ability to hide or explore call chain child elements taking less than 5% time of parent or with single child tiers.
- Mouse-over and printing of line of code as we browse through the CPU footprint distribution for hotspots or normal areas.

Can I do memory profiling with StrongOps ?

Yes, this is a newly introduced feature very similar to CPU profiling that enables you to profile memory footprint of an application, modules, functions and line of code and pinpoint bottleneck areas.

You can profile memory for a host and process over a period of time and identify the memory hotspots in code.

What is application tracing? Can I monitor my transactions?

StrongOps application tracing displays the path, sequence and time end-user transactions spend on the application components and endpoints. Application tracing provides visibility into the type of incoming HTTP request calls, backend systems being accessed, web-service calls and outbound HTTP responses the application makes to complete delivery of transactions.

This tracing is provided at the component level and aggregate transaction levels. We are actively working on deep-dive single transaction tracing across JavaScript, event and native levels for individual transaction instances.

Glossary

ACL

Access control list. Used to restrict users' and applications' access to data in models.

Adapters

Adapters provide the transport-specific mechanisms to make remote objects (and collections thereof) available over their transport.

cluster

A set of identical Node worker processes all receiving requests on the same port. See also: worker.

Connector

See LoopBack Connector.

CRUD

Create, read, update and delete (CRUD), the four basic functions of persistent storage typically provided by LoopBack data sources.

data source

A factory for model classes. A data source connects with specific database or other backend system using a connector. All model classes within single data source shares same connector type and one database connection. But it's possible to use more than one data source to connect to different databases.

Enterprise connector

Module the connects to backend data source such as Oracle, MySQL, or MongoDB.

hook

TBD. See [Strong Remoting](#).

LDL

LoopBack Definition Language, a simple way to define LoopBack data models in JavaScript or JSON. LDL compiles a model definition into a JavaScript constructor.

libuv

Multi-platform support library with focus on asynchronous I/O, primarily developed for use by Node.js.

LoopBack

Mobile backend framework that can run in the cloud or on your own servers. Built on StrongNode and open-source Node.js modules.

LoopBack connector

A LoopBack connector provides access to a backend system such as a database, REST API, or other service. Connectors are not used directly by application code. Rather, you use the LoopBack Datasource Juggler to create a dataSource to interact with the connector.

LoopBack DataSource Juggler

An object-relational mapping that provides a common set of interfaces for interacting with databases, REST APIs, and other data sources.

MBaaS

Mobile backend as a service. See [Backend as a service](#) on Wikipedia.

model

A LoopBack Model consists of: Application data Validation rules Data access capabilities Business logic A model provides a remote API that clients use to display user interface elements and to interact with backend systems.

Node.js

Software platform used to build scalable network applications. Node.js uses JavaScript as its scripting language, and achieves high throughput via non-blocking I/O and a single-threaded event loop. See Node.js on Wikipedia. Usage note: Initially, "Node.js," thereafter "Node."

NodeFly

See [StrongOps](#).

npm

Node package manager, the command-line tool for installing applications and managing dependencies using the npm registry.

On-Premises

Located in the customer's own datacenter. Note ending "s."

open-source, open source

When used as an adjective, hyphenate; for example: "This is open-source software." See Open-source software on Wikipedia. Note: Although it is common not to hyphenate this term, we are using the standard English rules for hyphenating a compound adjective.

push notification

A short text message or other notification sent from a server application to client mobile apps. See [Creating push notifications](#).

remote objects

JavaScript objects exported by a StrongLoop app over the network in the same way you export functions from a module. You can invoke methods on remote objects locally in JavaScript

runtime

At the time of application execution, in contrast to when the application is compiled or deployed. Do not hyphenate.

slc

StrongLoop Control, the command-line tool for StrongLoop Suite development and operations. It provides command line access to all StrongLoop facilities. For more information, see [Working with standard Node apps](#).

streams

See [Strong Remoting](#).

StrongLoop Suite

An suite of tools and APIs for connecting Enterprise data to devices and browsers using Node.js technology.

StrongNode

StrongLoop's enterprise Node.js package containing a set of commonly-used modules known as "userland" modules and a comprehensive toolset intended for production use.

StrongOps

Web-based control center that enables you to visualize Node application performance, maintain your application in production, and optimize its performance.

Worker

A node child process.