The Code Barbarian

The software development blog of Valeri Karpov

Introduction to the MEAN Stack, Part One: Setting Up Your Tools

Posted on July 22, 2013 by vkarpov15
I've received several emails asking for instructions on how to set up a basic MEAN stack app. I'm going to take it one step further and give you guys a two-part post that will walk you through creating your first MEAN stack app- from installing the tools to actually writing the code. In Part One we'll go through the setup and installation process. Next in Part Two we'll walk through the steps for building a very simple to-do list. Part One consists of seven steps, although only the first two are are strictly necessary.

We'll start by installing all of our tools. NodeJS and MongoDB are designed to be as close to operating system independent as possible and we will be covering three of the most common operating systems here – OSX, Windows 7/8, and Ubuntu. The one tool that you'll need to get started is a *bash shell* (the standard linux command line). This tool takes different names on different operating systems, but they are all effectively the same for the purposes of this tutorial. If I use the term terminal, shell, or command line, I'm referring to a bash shell window. If you're on a Mac or an Ubuntu machine, you're looking for the Terminal. Windows doesn't have one by default, but there are several alternatives. The git installer for Windows (http://msysgit.github.io/) comes with "git bash," which is the tool that I currently prefer.

If you're on OSX, I highly recommend you install Brew (http://mxcl.github.io/homebrew/) to help with this process. Brew is a phenomenally useful tool for installing programs directly from your command line. For example, if you wanted to install git the hard way, you would have to open up your browser, google "git," click on a few links, download an installer, and run it. With brew, you would simply open up the Terminal, type *brew install git* , hit enter, and you're done. Brew will help a lot with setting up MongoDB and NodeJS.

1) Installing MongoDB-
First, we're going to install MongoDB.
OSX: Open up your Terminal window and run

```
sudo brew install mongodb
```

Ubuntu: Open up your shell and run:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart (http://downloads-d
sudo apt-get update
sudo apt-get install mongodb-10gen
```

Windows: Go to http://www.mongodb.org/downloads (http://www.mongodb.org/downloads) and download the latest version of MongoDB for Windows, which should be a plain zip file. Once the download is done, extract the contents of the zip file to a memorable directory, such as 'c:\mongodb'.

That's it! After the installation process is done, you should be able to run the command

```
mongod
```

directly from your command line on Mac and Ubuntu. On Windows, you'll need to run it as

```
/c/mongodb/bin/mongod
```

Think of mongod as your local MongoDB server- you need to have a mongod process running in order to be able to use MongoDB. You can now leave mongod running, open up another terminal, and run

```
mongo test
```

or

```
/c/mongodb/bin/mongo test
```

on Windows. This should open up the MongoDB shell. Hit ctrl-c to close the MongoDB shell.

2) Installing NodeJS and npm-
Next we're going to install NodeJS and npm (node package manager).
Mac: Open your terminal and run

```
sudo brew install node
curl http://npmjs.org/install.sh | sh
```

(Instructions from http://madebyhoundstooth.com/blog/install-node-with-homebrew-on-os-x/ (http://madebyhoundstooth.com/blog/install-node-with-homebrew-on-os-x/))

Ubuntu:

```
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 npm
```

Windows: Download the installer from http://nodejs.org/download/ (http://nodejs.org/download/). I recommend using the installer rather than the binary to save yourself the extra work of adding the binary location to the system path.

Once you have successfully installed NodeJS, you should be able to run the "node" and "npm" commands from your terminal. When you run "node," you should see a single ">." This is the node shell, hit ctrl-c to close it.

When you run "npm" with no arguments, you should see a bunch of usage information. Keep in mind that "npm" often requires root permissions to run. If an npm command fails for unclear reasons, try running it through "sudo." (http://xkcd.com/149/)

At this point, you've done essentially all you need to do to run a MEAN stack application. You can simply clone/fork https://github.com/vkarpov15/mean-stack-skeleton (https://github.com/vkarpov15/mean-stack-skeleton) , start a "mongod" process, navigate to the git repo, and run

```
npm install -d
```

You should then be able to run

```
node app.js
```

to start your server. However, I recommend at least reading through the rest of the below steps, as they will explain in a bit more detail some of the tools that you will be using. You can follow along step by step from the mean-stack-skeleton repo commit log (https://github.com/vkarpov15/mean-stack-skeleton/commits/master), because, conveniently, the commits correspond one-to-one with steps 4-7.

3) Installing ExpressJS-
Now that we have MongoDB and NodeJS, it is time to install ExpressJS- this is pretty trivial now that we already have npm installed from step 2. In Mac, Linux, and Windows, you should open up a terminal and run

```
npm install express -g
```

The "-g" flag means that the package will be installed so you can run it from your terminal.

We're using ExpressJS here because it adds some extremely useful tools for web development that NodeJS lacks. Contrary to what you may have heard, NodeJS is not a fully featured web server. NodeJS is simply a tool for doing I/O in an event-based concurrency framework with Javascript. It may be sufficient for a trivial server that prints "Hello, World" for every HTTP request, but it makes more sophisticated web development more difficult than it has to be. ExpressJS provides a familiar MVC (Model-View-Controller) framework for you to work with.

4) Creating an ExpressJS application-
Now that we have all of our server tools in place, let's create an ExpressJS application. Start by running the command

```
express mytestapp
cd mytestapp
npm install
```

This should create a "mytestapp" folder in your current directory that will contain your application. You should be able to now run

```
node app.js
```

from the "mytestapp" folder. If you point your browser to http://localhost:3000 (http://localhost:3000), you should see a simple "Welcome to Express" screen.

The "mytestapp" directory will contain a few sub-directories: *routes*, *views*, *public*, and *node_modules*, as well as *app.js* and *package.json* files. Here's a brief description of what these files and directories do:

- *app.js*: The main point of entry into your web server. This file defines the port that your application listens for requests on, includes dependencies via the "require" function, and defines handler functions for different REST-ful paths your clients can access.
- *package.json*: Defines internal dependencies for your application. Running "npm install -d" (which we will do shortly when we modify this file) installs all the dependencies listed in the "dependencies" file.
- *routes*: The routes directory will contain Javascript handlers for REST-ful paths defined in app.js. For example, if you open index.js, you'll see the handler for requests to the "/" path, which renders the "index" template that resides in "views/index.jade".
- *views*: The views directory will contain templates defined in the Jade language. Jade is a cleaner and more human-readable version of HTML with several extraordinarily useful features, such as inheritance and partials. Your routes will access these views via the "res.render" function that you saw in routes/index.js.
- *public*: The public directory is usually used for images, client-side Javascript, and other static assets. ExpressJS will route requests that correspond to files under the public directory directly to the file. For

example, if you run

```
node app.js
```

and point your browser to http://localhost:3000/stylesheets/style.css (http://localhost:3000/stylesheets/style.css), you'll see that Express returned the contents of the "public/stylesheets/style.css" file.

- *node_modules*: The node_modules file contains source code for tools installed via npm. You can feel safe ignoring its contents for now.

5) Installing drivers and MongooseJS-
Now we're going to install the MongoDB driver for NodeJS, as well as another useful tool called MongooseJS. MongoDB is organized with language-specific drivers that interface with the core database, so in order to use MongoDB with NodeJS, we need the NodeJS driver. MongooseJS is an ORM for NodeJS and MongoDB that is relatively similar to Ruby on Rails' ActiveRecord. For more discussion of MongooseJS, see my original introductory post about the MEAN Stack (http://thecodebarbarian.wordpress.com/2013/04/29/easy-web-prototyping-with-mongodb-and-nodejs/), as well as the MongooseJS-specific follow-up Mistakes You're Probably Making With MongooseJS, And How To Fix Them (http://thecodebarbarian.wordpress.com/2013/06/06/61/).

Open up the package.json file in "mytestapp" using your text editor of choice. I prefer SublimeText (http://www.sublimetext.com/) on Mac and Windows, and gedit (https://projects.gnome.org/gedit/) or vim (http://www.vim.org/) on Linux, but anything like Notepad or Textmate should work fine. The package.json file should contain a list of dependencies which looks like this:

```
"dependencies": {
  "express": "3.0.3",
  "jade": "*"
}
```

We're going to add two more lines

```
"dependencies": {
  "express": "3.0.3",
  "jade": "*",
  "mongodb": ">= 0.9.6-7",
  "mongoose" : ">= 3.6"
}
```

Next run

```
npm install -d
```

If you're on OSX or Ubuntu, you may have to run "sudo npm install -d" to get it working. Now that we have both the NodeJS MongoDB driver and MongoDB, we can create a connection to MongoDB via Mongoose in our app.js file:

```
var Mongoose = require('mongoose');
var db = Mongoose.createConnection('localhost', 'mytestapp');
```

You can see the diff for this step on github here (https://github.com/vkarpov15/mean-stack-skeleton/commit/2eec604f6698ee99d9d3d88070835073c0e8e0c6). Run "node app.js" now – your "mongod" process should output a message that looks something like "[initandlisten] connection accepted from 127.0.0.1," which indicates that your app has successfully connected to your mongod process.

6) Use Bower for adding AngularJS-
Bower is another npm tool that we will use to add AngularJS to our app. It is essentially an npm for client-side Javascript. For example, if you want to add jQuery to your project, Bower enables you to do so from your terminal with the command "bower install jquery." Install it with

```
1 | npm install bower -g
```

from your terminal.

Bower has one minor quirk that you need to be aware of – it will install components into a "bower_components" directory by default, which unfortunately is not under the public directory. From your "mytestapp" directory, create a directory called "vendor" under "public/javascripts." You can do this by running the command "mkdir public/javascripts/vendor." Next, in the "mytestapp" directory, create a plain text file called ".bowerrc" that contains:

```
1 | { "directory" : "public/javascripts/vendor" }
```

This configuration file will tell Bower to install tools into the "public/javascripts/vendor" directory. You can see the diff for this step on github here (https://github.com/vkarpov15/mean-stack-skeleton/commit/35afa3e3c11bb2cf100b294eb74adf4158446755).

7) Installing AngularJS-
Now lets install AngularJS v1.0.6 using "bower install angular#1.0.6." You should see a *public/javascripts/vendor/angular* directory that contains "angular.js," "angular.min.js," and "bower.json" once you're done.

To be continued
Congratulations, you've now officially completed Part One and set up the MEAN Stack on your machine! Part Two will be posted in a week, and in it we'll go step-by-step for building a simple to-do list application using the MEAN Sack. In the MEANtime (pun very much intended), I encourage you to play around with these tools and see if you can build anything cool with them. Could you whip up a sweet little MEAN Stack application before I get a chance to drop the next part of the guide? I certainly think you guys can figure it out. It's always great to hear about the awesome ways people are using the MEAN Stack, so if you decide to make something please go ahead and share it with everybody in the comments. Maybe it will turn into a real product and you'll make billions of dollars and commission the construction of a solid gold yacht despite the insistence of your engineers that it will totally sink but you build it anyway because you're so rich it doesn't matter if it ends up on the bottom of the ocean, laws of physics be damned. Or maybe the other readers will just say "sweet app bro." Either way, what are you waiting for?!

Want to learn more in advance of Part Two? Howtonode.org has a good tutorial on how to use the NodeJS MongoDB driver and ExpressJS to build a simple blogging platform at http://howtonode.org/express-mongodb (http://howtonode.org/express-mongodb). In a previous post (http://thecodebarbarian.wordpress.com/2013/05/12/how-to-easily-validate-any-form-ever-using-angularjs/) I described how integrate MongooseJS validation and AngularJS, which will be useful for the next post. Thanks as always to my partner William Kelly (@idostartups) for his help writing this post, and for getting MEAN Stack shared across all of the internets.

Bookmark the permalink.

63 thoughts on “Introduction to the MEAN Stack, Part One: Setting Up Your Tools”

Stoto says:
on July 23, 2013 at 9:28 am
Hey,
This is what i did with the mean stack:

http://bitcoin.stoto.net/

NodeJS, HJS, angular, mongodb, it’s awsome, simple and fun to develop with.

Reply

vkarpov15 says:
on July 23, 2013 at 3:38 pm
Love it. Bitcoins are awesome. Also a huge fan of how you used socket io on the client for connecting to the Mt Gox streaming API. Btw, did you use a charting API?

Reply

Stoto says:
on July 25, 2013 at 7:44 am
I used highchart for the charts with data backed from mongodb.

I also used socket.io to trigger chart reload when there is new data available in the backend.

2. Pingback: Introduction to the MEAN Stack, Part Two: Building and Testing a To-do List | The Code Barbarian

Chris Moudy says:
on August 1, 2013 at 9:51 pm
Loved the stack so much ended up dropping a vagrant kickstart project out in GitHub. Clone the repo, vagrant up the vms, start the express server and boom you have your base app ready to develop MEAN.

https://github.com/cmoudy/mean-vagrant

Reply

vkarpov15 says:
on August 1, 2013 at 10:30 pm
Love it. Thanks for pointing me to Vagrant, this is the first I've heard of it, looks pretty sweet.

Reply

Chris Moudy says:
on August 1, 2013 at 11:31 pm
Yeah Vagrant is solid and makes it so easy to be able to drop consistent environments. Combining that with your tutorial here seemed like a double win as then folks can get up and running in minutes (depending on download speed of course).

Dustin Davis (@DustinDavis) says:
on August 2, 2013 at 12:03 pm
Excellent tutorial. Thank you so much!

Reply

Nilgai says:
on August 13, 2013 at 1:36 pm
Really enjoying this two-part tutorial. Would like to try out it from Windows using WebMatrix3 and wondered what the implications would be in relation to setting up all the environment and tools (addressed in Part 1). If WM3 not suitable would consider trying webstorm – any comment?

Reply

Dan Sevush says:
on August 17, 2013 at 8:45 pm
I'm just past the installation and I look forward to the rest of the tutorial. I'm fairly new to using OSX as a developer, but I go back to the daze when unix required you log in remotely, so I'm pretty comfortable with a terminal.

Thanks so much for pointing to the command line tools and brew. I've gotten spoiled by ubuntu's package installer, so this is a welcome addition. And now I feel like I'm out of version/config hell and on to doing what I do best – bringing the machine to its knees.

I hit a few stumbling blocks. Some google-foo got me through it, but thought you might want to update your text.

Changing http: to https: helped me and others get past an error. There was a note in the page you reference
curl http://npmjs.org/install.sh | sh

Please advise people to enter hyphens manually, as they appear to get mangled in copy/paste:
npm install express –g
did not work for me. I googled the error and found this helpful post -
https://github.com/isaacs/npm/issues/1859
"Manually type "-g", wherever you copied the command line from used an en dash character rather than a hyphen."

And I'm not sure how much this matters, but I installed brew under my user name. You have a lot of references to sudo, which did not work for me and in fact, everything worked fine. So I think the basic premise is missing, which is IF you choose install brew as root, then sudo would apply for subsequent steps.

I'm one of those people who prefers to be a user and sudo or su only when needed. I suspect I'm going to be fine using brew and the installed packages as a user, but you should definitely clarify the difference at the start.

Reply

Dan Sevush says:
on August 17, 2013 at 8:55 pm
This is the error I got when attempting to follow your instructions to use sudo. Should I have installed brew differently?
Dans-MacBook-Pro:~ dansevush$ sudo brew install mongodb
Error: Cowardly refusing to `sudo brew install`
You can use brew with sudo, but only if the brew executable is owned by root.
However, this is both not recommended and completely unsupported so do so at
your own risk.

Reply

vkarpov15 says:
on August 19, 2013 at 3:14 am
Hmm that's interesting. Maybe I was a bit sloppy with the Mac instructions. Thanks for finding this. Have you tried running it without sudo?

vkarpov15 says:
on August 19, 2013 at 3:15 am
My apologies about that, at some point the text must've gotten converted by Microsoft Word or OpenOffice into the hyphen character. Thanks for the catch.

Reply

Dan Sevush says:
on August 20, 2013 at 3:23 am
I've been running into small problems almost every step of the way. Sudo is definitely not needed, and I've got one last error. When I attempt to install angular, I get this error:
Error: Unable to parse /Users/dansevush/testapp/.bowerrc: Unexpected token "
I have tried removing all white space, inserting and removing line feeds to no avail.
It would be really sweet if there was something equivalent to LAMP, WAMP, etc. that handles all package installations in one fell swoop

Dan Sevush says:
on August 20, 2013 at 3:38 am
Ah, another Word-quotification mishap. It's very subtle, but the quote character is different. I typed the line over by hand and was able to install angular. on to the next part

My suggestion going forward – just edit (or re-edit) every line using plain text tools. We don't need no stinkin' MSOffice apps

Reply

Jeff Helman says:
on December 29, 2013 at 7:38 pm
VERY helpful comment. Thanks! FYI, this happened to me using a Mac too. In my case, it may have been an errant trailing blank space when I did the copy/paste from the post.

Reply

vkarpov15 says:
on December 30, 2013 at 4:25 pm
Thanks for pointing this out guys. I've gone through and all of these little formatting weirdnesses should be fixed. Let me know if something else is amiss.

jshorwitz says:

on August 27, 2013 at 8:22 pm
I'm running this on an EC2 Ubuntu Server: ec2-50-18-42-166.us-west-1.compute.amazonaws.com and not sure what to enter for the mongodb at step 5:
2

var Mongoose = require('mongoose');
var db = Mongoose.createConnection('localhost', 'mytestapp');

Reply
Chad Lonberger says:
on August 27, 2013 at 11:38 pm
Strongly recommend a link to this tutorial is added to the github page… had numerous problems with the install due to some issues with my dev environment and this would have helped significantly. Thanks for writing it.

Reply
vkarpov15 says:
on October 31, 2013 at 9:13 pm
Good idea, I'll make sure to do this. Thanks for the suggestion.

Reply
Austen C. says:
on September 3, 2013 at 11:46 pm
I tried following your tutorial exactly and found after I ran 'node app.js' it was missing 'express' (and many other things like 'jade') — running 'npm install' from within the mytestapp directory installed all needed dependencies. I'm on win7 x64 using git bash, cheers!

Reply
vkarpov15 says:
on October 31, 2013 at 9:12 pm
Hmmm I could've sworn I put instructions to do npm install a few times in the tutorial. I'll look through and see if I can make that more clear. Thanks for sharing your troubles.

Reply
mp says:
on October 31, 2013 at 9:03 pm
I think this is a great way to use the express web framework but why would you use that if you ultimately plan to use Angular? Angular has it's own directory structure and routing methodology. Seems redundant and likely to have people confused.

Reply
vkarpov15 says:
on October 31, 2013 at 9:11 pm
Good comment. AngularJS does have its own methodology for routing, but only if you choose to use the routing functionality, which is not necessarily the right choice for all applications. Furthermore, even if you use AngularJS routing for all your views, you will almost certainly need some sort of RESTful API to your server, and NodeJS and ExpressJS make writing very sophisticated REST APIs very simple.

Reply

12. Pingback: How do I name the .bowerrc file?CopyQuery CopyQuery | Question & Answer Tool for your Technical Queries,CopyQuery, ejjuit, query, copyquery, copyquery.com, android doubt, ios question, sql query, sqlite query, nodejsquery, dns query, update query, ins

vkarpov15 says:

on November 25, 2013 at 8:34 pm
.bowerrc is a perfectly valid filename, even on Windows. Use git bash or whatever your Windows bash shell of choice is and create it from there if Windows Explorer won't let you do it.

Reply

slemn3 says:
on December 1, 2013 at 10:35 pm
Similar to "-g" issue, I think it is worth mentioning that if you copy paste the contents of the .bowerrc file, it will add the wrong double quote character. Those need to be typed as well.

13. Pingback: server setup | VolosHack

Robert says:
on December 9, 2013 at 6:46 am
to create .whatever files in windows from explorer, type .whatever. (with a trailing dot). When you click enter, Explorer will rename as simply .whatever

Reply

pigiuz says:
on December 12, 2013 at 9:49 am
when getting problems with npm install requiring you to sudo DON'T DO IT!
just change folder owner
sudo chown -R $USER /usr/local
anyone can deploy whatever, don't make it easy to grant root permissions to malicious scripts

Reply

vkarpov15 says:
on December 12, 2013 at 2:52 pm
That's a pretty good point. But with or without root privileges, a malicious npm module can do some real damage to your computer (e.g. steal your private key, wipe out your home directory, etc.). I've never had this problem, but right now the unfortunate truth of npm is that its up to you to either trust the npm module, scan through it yourself, or make sure you're running node on a user that can't access any files you find particularly valuable.

Reply

pigiuz says:
on December 12, 2013 at 3:57 pm
true, if you're not as paranoid as I am maybe the point of not sudoing could be just avoiding to type the password everytime you want to install a package

16. Pingback: Nodejs and the MEAN stack for the .NET developer part 3 | The Buttonfactory

sayth renshaw (@flebberX) says:
on January 1, 2014 at 9:28 am
I am receiving this error when going to localhost:3000.

Welcome to #{title} `doctype 5` is deprecated, you must now use `doctype html`

Reply

vkarpov15 says:
on January 17, 2014 at 9:26 pm
Thanks for the heads up. There's an issue open on github, I'll tweak that this weekend =)

Reply

Nanjiro says:
on January 17, 2014 at 5:23 pm
Step 4 on windows
if you run | node app.js
I got an error: Cannot find module 'express'
So you need to run the "npm install" first

Reply

vkarpov15 says:
on January 17, 2014 at 9:28 pm
I believe that `npm install` is in step 2.

Reply

vkarpov15 says:
on January 17, 2014 at 9:29 pm
Excellent name, by the way. Any relation to the Prince of Tennis character?

Hollister says:
on January 20, 2014 at 4:45 pm
I'd like to use Brunch to build the app. It moves the compiled files from app to public. Any suggestions on how to change the folder structure to accommodate this?

Reply

vkarpov15 says:
on January 21, 2014 at 3:44 pm
This is the first I've heard of Brunch, so I'll have to investigate a bit more. Brunch does look very useful though, thanks for the heads up.

Reply

Watermelon says:
on February 5, 2014 at 8:11 am
I'm running into an issue on step 5. I've added
var Mongoose = require('mongoose');
var db = Mongoose.createConnection('localhost', 'mytestapp');
and I do get "[initandlisten] connection accepted from 127.0.0.1,"
but I also noticed the following:
Failed to load c++ bson extension, using pure JS version
Express server listening on port 3000
Any thoughts on how to get rid of/fix this?

Reply

vkarpov15 says:
on February 7, 2014 at 6:30 pm
I've only seen this happen on Windows, its a slight wonkiness with npm and the NodeJS MongoDB driver. There's some ideas as to how to fix it here: https://groups.google.com/forum/#!topic/node-mongodb-native/db4xD0x42eM

Reply

Tony says:
on February 25, 2014 at 2:10 pm
using sudo with brew is a no no

Reply

afdga says:
on March 4, 2014 at 5:50 pm
I know I'm late to the party , but shouldn't we do another "npm install" after doing "express mytestapp" and "cd mytestapp"

Reply

vkarpov15 says:
on March 6, 2014 at 7:53 pm
Good call, thanks. Fixed that.

Reply

Roseanne says:
on March 8, 2014 at 3:49 am
Just wish to say your article is as surprising. The clarity in your post is simply excellent and i can assume you're an expert on this subject.
Fine with your permission allow me to grab your feed to keep up to date with forthcoming post. Thanks a million and please carry on the enjoyable work.

Reply

24. Pingback: Installing and Configuring the MEAN Stack, Yeoman, and Associated Tooling on Windows | Programmatic Ponderings

jimcal says:
on March 15, 2014 at 6:26 pm
Hi Vkarpov15,

thanks for introducing this and I want to follow up with a question,

I was following the steps on my Mac, up until step 5 everything was good.

And I put mongodb and mongoose into app.js, ran "node app.js" and this happened, any chance you recognize what I may have missed?

Express server listening on port 3000

events.js:72
throw er; // Unhandled 'error' event
^
Error: failed to connect to [localhost:27017]
at null. (/Users/jimcal/mytestapp/node_modules/mongodb/lib/mongodb/connection/server.js:553:74)
at EventEmitter.emit (events.js:106:17)
at null.
(/Users/jimcal/mytestapp/node_modules/mongodb/lib/mongodb/connection/connection_pool.js:140:15)
at EventEmitter.emit (events.js:98:17)
at Socket.
(/Users/jimcal/mytestapp/node_modules/mongodb/lib/mongodb/connection/connection.js:512:10)
at Socket.EventEmitter.emit (events.js:95:17)
at net.js:441:14
at process._tickCallback (node.js:415:13)

Reply

vkarpov15 says:
on March 28, 2014 at 6:11 pm

The error says that Node can't connect to the MongoDB server, which runs on port 27017 by default. Make sure you have a "mongod" process running, run `ps -ef | grep "mongod"`, you should see at least 2 rows. If not, MongoDB has a reasonable guide about the different options in starting the "mongod" process: http://docs.mongodb.org/manual/tutorial/manage-mongodb-processes/

Reply

nickb says:
on April 3, 2014 at 12:16 am
FYI, not sure if this is Win7-specific, but at the end of Step 2 where you say you should be able to run "node app.js", it requires Express to be installed first to work (step 3). Thanks for the tutorial, though!

Reply

Hans VB says:
on April 10, 2014 at 11:11 pm
Trying this on Ubuntu Saucy, I was missing the (very handy!) express binary, which creates your project structure. I didn't want to use the older express version that can be installed with 'apt-get install node-express' (it creates a not so nice and not so current project structure). Reading the Readme.md from express, I finally learned the newest version can easily be installed with 'npm install -g express-generator'.

You also might want to add the apt-repository https://launchpad.net/~chris-lea/+archive/node.js/ for newer nodejs (and npm) versions on Ubuntu.

Reply

Trent says:
on May 12, 2014 at 4:15 am
"npm install express -g" didn't work for me in windows, I had to use "npm install -g express-generator"

Reply

vkarpov15 says:
on June 3, 2014 at 4:23 pm
Interesting, its always worked fine for me. What version of express did you install?

Reply

sarvesh says:
on August 23, 2014 at 7:17 am
I too faced the same issue, with installing 'express'. Found a question in stack overflow regarding this.
http://stackoverflow.com/questions/23097826/install-express-with-npm
May be, this can be added (for express 4)in the tutorial .

29. Pingback: MEAN Stack | Evan McCullough

Elijah says:
on May 28, 2014 at 9:46 pm
My coder is trying to persuade me to move to .net from PHP.
I have always disliked the idea because of the costs.

But he's tryiong none the less. I've been using WordPress on several
websites for about a year and am worried about switching to another platform.

I have heard good things about blogengine.net. Is there a way I can transfer
all my wordpress posts into it? Any help would be really appreciated!

Reply

Nate says:

on June 9, 2014 at 3:15 pm
I seem to be the only one running into problems at step 1. I'm running in Ubuntu 12.04. When running the first command (sudo apt-key adv –keyserver keyserver.ubuntu.com –recv 7F0CEB10) I think it's working, though part of the response says "no ultimately trusted keys found." Then, I can't figure out why there is no closing single quote (') in the echo command. I went ahead and ^C out of it and tried the third command (sudo apt-get update) which didn't give me an error, somehow (considering I didn't properly follow the second command). So I went ahead and tried the fourth command, at which point I finally got the error, "Unable to locate package mongodb-10gen."
So, perhaps I tried to much after the first question… I'm not really sure which part went wrong first.

Reply
seansean11 says:
on June 16, 2014 at 9:28 pm
Great install tutorial! Thanks for walking me through the different steps of setting up the mean stack instead of just having us install the mean-stack-skeleton. It helped me understand the anatomy of the Mean stack and setting up the application

Reply
33. Pingback: "GPG Error: Clearsigned file isnt valid" When Installing Mongodb - Tech Forum Network

dani0010 (@dani0010) says:
on July 14, 2014 at 12:42 pm
Thank you so much for this article, but two small things have changed that break these instructions. As someone completely new to MEAN, I spent several hours banging my head against a wall only to discover it actually wasn't me!

1) As a previous commenter mentioned, in addition to the npm install -g express command, you must also run npm install -g express-generator.
2) To start node, you should run npm start from your app root. When I ran node app.js nothing happened.

Reply
vina melody (@vinamelody) says:
on August 6, 2014 at 3:21 pm
Do i need to do all these if i am installing mean.io ? Thanks!

Reply
vkarpov15 says:
on September 4, 2014 at 9:39 pm
Don't know much about mean.io, this is completely unrelated.

Reply
ben says:
on August 24, 2014 at 12:16 pm
when trying to install node.js and npm on linux mint, i got
nodejs : Conflicts: npm
E: Unable to correct problems, you have held broken packages.
turns out node.js in chris lea repository comes with npm…

Reply

Follow

Follow “The Code Barbarian”