Migrating to Meteor 1.4

Breaking changes

These are all the *breaking changes* — that is, changes you absolutely have to worry about if you are updating your app from 1.3.x to 1.4. However, we recommend that you also consider the *recommended* changes listed below.

The `babel-runtime` npm is usually required

The `babel-runtime` npm package is generally required as a dependency since the Meteor `babel-runtime` package no longer attempts to provide custom implementations of Babel helper functions. To install the `babel-runtime` npm, run the following command in any Meteor application:

```
meteor npm install --save babel-runtime
```

New projects created with `meteor create <app-name>` will automatically have this package added to their `package.json`'s `dependencies`.

Binary Packages require a Build Toolchain

The headline feature of Meteor 1.4 is the upgrade to Node version 4. Node 4 includes a changed ABI (application binary interface), which means that *binary npm packages* that your application uses will need to be recompiled.

Some very common binary packages (such as `npm-bcrypt`) will already have been republished for the Node 4 platform, so if you are using a limited set of packages, this may not affect you; however if you are using less common dependencies, this may be an issue.

If you have binary npm packages in your application `node_modules` directory, you should run `meteor npm rebuild` (after `meteor update`) in your application directory to recompile those packages.

Meteor will automatically recompile any binary npm dependencies of Meteor packages, if they were not already compiled with the correct ABI. This will typically happen the first time you start your application after updating to 1.4, but it may also happen when you `meteor add some:package` that was published using a different version of Meteor and/or Node.

In order for this rebuilding to work, you will need to install a basic compiler toolchain on your development machine. Specifically,

- OS X users should install the commandline tools (in short, run `xcode-select --install`).
- Windows users should install the MS Build Tools.
- Linux users should ensure they have Python 2.7, `make` and a C compiler (g++) installed.

To test that your compiler toolchain is installed and working properly, try installing any binary npm package in your application using `meteor npm`. For example, run `meteor npm install bcrypt` then `meteor node`, then try calling `require("bcrypt")` from the Node shell.

Update from MongoDB 2.4

Meteor has been updated to use version 2.2.4 of the node MongoDB driver. This means Meteor now ships with full support for MongoDB 3.2 (the latest stable version) and the WiredTiger storage engine. We recommend you update your application to MongoDB 3.2.

If you are currently using MongoDB 2.4, please note that the version has reached end-of-life and you should at the least update to version 2.6. Version 2.6 is the minimum version supported by Meteor 1.4.

Updating your database to 2.6 is generally pretty painless. Please consult the MongoDB documentation for details about how to do so.

> As of 1.4, you must ensure your `MONGO_OPLOG_URL` contains a `replicaSet` argument (see the changelog and the oplog documentation).
>
> NOTE: Some MongoDB hosting providers may have a deployment setup that doesn't require you to use a `replicaSet` argument. For example, Compose.io has two types of deployments, MongoDB Classic and MongoDB+. The new MongoDB+ offering is a sharded setup and not a true replica set (despite the shard being implemented as a replica set) so it does not require the `replicaSet` parameter and Meteor will throw an error if you add it to your connection strings.
>
> If you see a failed authentication you may need to upgrade to SCRAM-SHA-1, essentially: `use admin, db.adminCommand({authSchemaUpgrade: 1});`. You may need to delete and re-add your oplog reader user.

Remove debugger statements

Due to changes in Node 4, if you have `debugger` statements in your code they will now hit the breakpoint even without a debugger attached. This also means you can now debug without using the `--debug-brk` option.

Password reset and enrollment tokens now expire

Password Reset tokens now expire (after 3 days by default – can be modified via Accounts.config({ passwordResetTokenExpirationInDays: ... }). PR #7534

See PR #7794 for infomation about splitting reset vs enrollment tokens and allowing different expiration times.

Recommendations

Update to Meteor 1.3.5.1 first

Though not mandatory, it may be helpful to update your apps to Meteor 1.3.5.1 before updating to 1.4, since 1.3.5.1 is the most recent release before 1.4, and contains much of the same code as 1.4. To update an app to 1.3.5.1, run `meteor update --release 1.3.5.1` in the app directory. When you are confident the app is working correctly, `meteor update` will take you all the way to Meteor 1.4.

Update to MongoDB 3.2

Although Meteor 1.4 supports MongoDB 2.6 and up, as well as the older MMAPv1 storage engine, we recommend you update your database to use the new WiredTiger storage engine and use MongoDB 3.2.

To update your production database to version 3.2 you should follow the steps listed in the MongoDB documentation. To update your storage engine, you should ensure you follow the "Change Storage Engine to WiredTiger" instructions in the 3.0 upgrade documentation.

If you are using OS X or 64bit Linux, you can update your development database in a similar way (if you are running `meteor` as usual, you can connect to the development database at `localhost:3001/meteor`). However, if you are not concerned about the data in your development database, the easiest thing to do is to remove all local data (including your development database) with `meteor reset`. When you next start `meteor`, the database will be recreated with a 3.2 WiredTiger engine.

If you are using Windows or 32bit Linux, you can update your development database to 3.2, however it will continue to use the MMAPv1 storage engine, as the 32bit MongoDB binary does not support WiredTiger.

Use Nested Imports

Thanks to the use of the reify library, Meteor now fully supports nested `import` declarations in both application and package modules, whereas previously they were only allowed in application code:

```
if (Meteor.isClient) {
  import { symbol } from './client-only/file';
}
```

One place this is particularly useful is in test files that are only intended to run on the client or the server — you can now use `import` wherever you like, without

having to organize your tests in `client` or `server` directories.