Most of the new features in Meteor 2.6 are either applied directly behind the scenes (in a backwards compatible manner) or are opt-in. For a complete breakdown of the changes, please refer to the changelog.

The above being said, there are a few items that you should implement to have easier time in the future.

Cordova - Launch Screens for iOS

You are now able to use dark theme specific splash screens for both iOS and Android by passing an object `{src: 'light-image-src-here.png', srcDarkMode: 'dark-mode-src-here.png'}` to the corresponding key in `App.launchScreens`

Legacy launch screens keys for iOS on `App.launchScreens` are now deprecated in favor of new storyboard compliant keys PR #11797. This will drop the following keys we have:
`['iphone5','iphone6','iphone6p_portrait','iphone6p_landscape','iphoneX_portrait','iphoneX_landscape','ipad_portrait_2x','ipad_landsc`

To migrate replace the deprecated keys
`['iphone5','iphone6','iphone6p_portrait','iphone6p_landscape','iphoneX_portrait','iphoneX_landscape','ipad_portrait_2x','ipad_landsc`
with the corresponding new key:
`['ios_universal','ios_universal_3x','Default@2x~universal~comany','Default@2x~universal~comcom','Default@3x~universal~anycom','Defau`
and adapt necessary splash images to the new dimensions asked by Apple. You can get more info here.

MongoDB 5.0

Introduction

Meteor before 2.6 was supporting MongoDB Server 4.x, starting from this version we've upgraded to MongoDB Node.js driver from version 3.6 to 4.3.1 which supports MongoDB Server 5.x.

This change was necessary at the time of writing this guide (January 2022) as MongoDB Atlas is going to migrate automatically all the clusters in the plans Atlas M0 (Free Cluster), M2, and M5 to MongoDB 5.0 in February 2022, but this change would be necessary anyway as this is now the latest version of MongoDB Server. The migration in the M0, M2 and M5 is just a sign from MongoDB that they believe MongoDB 5.0 should be the version used by everybody as soon as possible.

If you are running on MongoDB Atlas and in one of these plans you have to run Meteor 2.6 in order to connect and interact with your MongoDB properly as your MongoDB is going to be upgraded to 5.0 in February. If you are not running at these plans, you can continue to use your MongoDB at previous versions and you can use previous versions of Meteor still without any problems.

An important note is that we have migrated everything supported by Meteor to be compatible with MongoDB 5.x and also MongoDB Node.js Driver 4.x but this doesn't include, as you should expect, what you do in your code or package using `rawCollection`. `rawCollection` is a way for Meteor to provide you the freedom to interact with MongoDB driver but that also comes with the responsibility to keep your code up-to-date with the version of the driver used by Meteor.

That said, we encourage everybody to run the latest version of Meteor as soon as possible as you can benefit from a new MongoDB driver and also other features that we are always adding to Meteor.

This version of Meteor is also compatible with previous version of MongoDB server, so you can continue using the latest Meteor without any issues even if you are not running MongoDB 5.x yet. You can check here which versions of MongoDB server the Node.js driver in the version 4.3.0 supports and as a consequence these are the versions of MongoDB server supported by Meteor 2.6 as well. In short, Meteor 2.6 supports these versions of MongoDB server: 5.1, 5.0, 4.4, 4.2, 4.0, 3.6.

Embedded MongoDB

If you are using Embedded MongoDB in your local environment you should run `meteor reset` in order to have your database working properly after this upgrade. `meteor reset` is going to remove all the data in your local database.

rawCollection()

If you depend on any method inside the rawCollection() object, you have to review every call with the new driver API here. Also make sure you check all the changes made to the driver here and here.

You can check an example applied to the "aggregate" function, which had changes to its API. The collection.rawCollection().aggregate() function doesn't expect a callback any more like in older versions, and aggregate().toArray() now returns a promise. For this specific case, we have written the fix in this PR.

If you are a user looking for errors that are showing in a custom package, please open an issue in the package owner repository so the maintainer can make the due changes.

Cursor.count()

`applySkipLimit` option for count() on find cursors is no longer supported. By default, this option will always be true. So, for example, let's say you have a collection with 50 documents and execute a `find` with a limit of 25:

```
const cursor = collection.find({}, { limit: 25 });
```

When you call `cursor.fetch()`, the result will be 25 documents, and `cursor.count()` will be 25. Whereas, in the previous version, `cursor.count()` would result in 50, in this case, and you would need to provide the option `applySkipLimit` to get the result 25.

Now, you'll need to create a new cursor, but this time not providing a limit, in order to get the number of all documents in your collection:

```
const cursorWithLimit = collection.find({}, { limit: 25 });
const cursorWithNoLimit = collection.find({});
// cursorWithLimit.fetch() => returns 25 documents
// cursorWithNoLimit.count() => returns 50
```

Remember that a `find` is a wrapper for the query, so creating two or more cursors in a row is totally fine and not slower at all.

Changes

Here is a list of the changes that we have made to Meteor core packages in order to make it compatible with MongoDB Node.js Driver 4.3.x, most of them are not going to affect you but we recommend that you test your application well before upgrading to the latest version of Meteor as we have made many changes on how Meteor interact with MongoDB.

- internal result of operations inside Node.js MongoDB driver have changed. If you are depending on rawCollection results (not only the effect inside the DB), please review the expected format as we have done here
- useUnifiedTopology is not an option anymore, it defaults to true.
- native parser is not an option anymore, it defaults to false in the mongo connection.
- poolSize not an option anymore, we are using max/minPoolSize for the same behavior on mongo connection.
- fields option is deprecated, we are maintaining a translation layer to "projection" field (now prefered) until the next minor version, where we will start showing alerts.
- _ensureIndex is now showing a deprecation message
- we are maintaining a translation layer for the new oplog format, so if you read or rely on any behavior of it please read our oplog_v2_converter.js code
- update/insert/remove behavior is maintained in the Meteor way, documented in our docs, but we are now using replaceOne/updateOne/updateMany internally. This is subject to changes in the API rewrite of MongoDB without Fibers AND if you are using rawCollection directly you have to review your methods otherwise you will see deprecation messages if you are still using the old mongodb style directly.
- waitForStepDownOnNonCommandShutdown=false is not needed anymore when spawning the mongodb process
- _synchronousCursor._dbCursor.operation is not an option anymore in the raw cursor from nodejs mongodb driver. If you want to access the options, use _synchronousCursor._dbCursor.(GETTERS) - for example, _synchronousCursor._dbCursor.readPreference.
- the default write preference for replica sets on mongo v5 is w:majority
- If you are using MongoDB inside a Docker container in your dev environment, you might need to append directConnection=true in your mongouri to avoid the new mongo driver Service Discovery feature

It's worth mentioning that if you are using the built-in MongoDB that Meteor provides to run your app locally, you need to perform a `meteor reset` on your app to be able to use the version 2.6.

Below we describe a few common cases in this migration:

1) Same version of MongoDB server

If you are not changing your MongoDB server version you don't need to change anything in your code, even if you are using `rawCollection` results, but as we did many changes on how Meteor interact with MongoDB in order to be compatible with the new driver we recommend that you test your application carefully before releasing to production with this Meteor version.

We've made many tests in real applications and also in our automatic tests suite, we believe we've fixed all the issues that we found along the way but Meteor interaction with MongoDB is so broad and open that maybe you have different use cases that could lead to different issues.

Again, we are not aware of any issues that were introduced by these changes, but it's important that you check your app behavior, especially if you have places where you believe you are not using MongoDB in a traditional way.

2) Migrating from MongoDB 4.x to MongoDB 5.x

As we have made many changes to Meteor core packages in this version we recommend the following steps to migrate:

- Upgrade your app to use Meteor 2.6 (meteor update --release 2.6) in a branch;
- Create a staging environment with MongoDB 5.x and your app environment using the branch created in the previous step;
  - If you are using MongoDB Atlas, in MongoDB Atlas we were not able to migrate to a free MongoDB 5.x instance, so we had to migrate to a paid cluster in order to test the app properly, maybe this will change after February 2022;
- Set up your staging database with MongoDB 5.x and restore your production data there, or populate this database in a way that you can reproduce the same cases as if you were in production;
- Run your app pointing your MONGO_URL to this new database, that is running MongoDB 5.x;
- Run your end-to-end tests in this environment. If you don't have a robust end-to-end test we recommend that you test your app manually to make sure everything is working properly.
- Once you have a stable end-to-end test (or manual test), you can consider that you are ready to run using MongoDB 5.x, so you can consider it as any other database version migration.

We are not aware of any issues that were introduced by the necessary changes to support MongoDB, but it's important that you check your app behavior, especially if you have places where you believe you are not using MongoDB in a traditional way.

For example, the MongoDB Oplog suffered a lot of changes, and we had to create a conversor to keep our oplog tail understanding the changes coming from the oplog, so if you have complex features and queries that depend on oplog (Meteor real-time diff system), you should review if your app is working properly especially in this part.

As MongoDB also removed a few deprecated methods that were used by Meteor we recommend testing all important operations of your application.

Migrating from a version older than 2.5?

If you're migrating from a version of Meteor older than Meteor 2.5, there may be important considerations not listed in this guide (which specifically covers 2.4 to 2.5). Please review the older migration guides for details:

- Migrating to Meteor 2.5 (from 2.4)
- Migrating to Meteor 2.4 (from 2.3)
- Migrating to Meteor 2.3 (from 2.2)
- Migrating to Meteor 2.2 (from 2.0)
- Migrating to Meteor 2.0 (from 1.12)
- Migrating to Meteor 1.12 (from 1.11)
- Migrating to Meteor 1.11 (from 1.10.2)
- Migrating to Meteor 1.10.2 (from 1.10)
- Migrating to Meteor 1.10 (from 1.9.3)
- Migrating to Meteor 1.9.3 (from 1.9)
- Migrating to Meteor 1.9 (from 1.8.3)
- Migrating to Meteor 1.8.3 (from 1.8.2)
- Migrating to Meteor 1.8.2 (from 1.8)

- Migrating to Meteor 1.8 (from 1.7)
- Migrating to Meteor 1.7 (from 1.6)
- Migrating to Meteor 1.6 (from 1.5)
- Migrating to Meteor 1.5 (from 1.4)
- Migrating to Meteor 1.4 (from 1.3)
- Migrating to Meteor 1.3 (from 1.2)