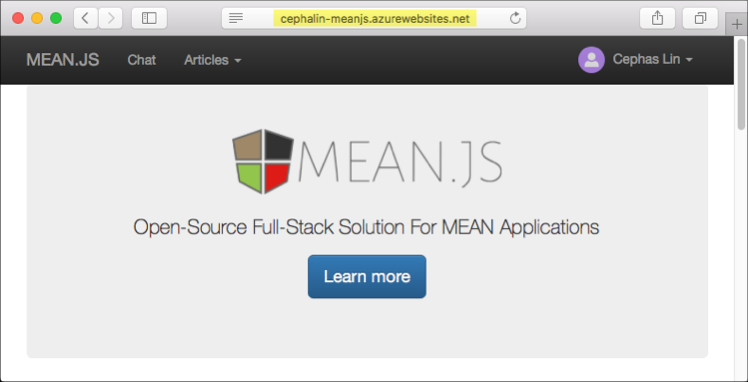
Azure Cosmos DB: Migrate an existing Node.js MongoDB web app

Azure Cosmos DB is Microsoft’s globally distributed multi-model database service. You can quickly create and query document, key/value, and graph databases, all of which benefit from the global distribution and horizontal scale capabilities at the core of Azure Cosmos DB.

This quickstart demonstrates how to use an existing [MongoDB](https://docs.microsoft.com/en-us/azure/cosmos-db/mongodb-introduction) app written in Node.js and connect it to your Azure Cosmos DB database, which supports MongoDB client connections. In other words, your Node.js application only knows that it's connecting to a database using MongoDB APIs. It is transparent to the application that the data is stored in Azure Cosmos DB.

When you are done, you will have a MEAN application (MongoDB, Express, Angular, and Node.js) running on [Azure Cosmos DB](https://azure.microsoft.com/services/cosmos-db/).



Launch Azure Cloud Shell

The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. Just click the **Copy** to copy the code, paste it into the Cloud Shell, and then press enter to run it. There are two ways to launch the Cloud Shell:

|  |  |
| --- | --- |
| Click **Try It** in the upper right corner of a code block. | Cloud Shell in this article |
| Click the **Cloud Shell** button on the menu in the upper right of the [Azure portal](https://portal.azure.com/). | Cloud Shell in the portal |
|  |  |

If you choose to install and use the CLI locally, this topic requires that you are running the Azure CLI version 2.0 or later. Run az --version to find the version. If you need to install or upgrade, see [Install Azure CLI 2.0](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli).

Prerequisites

If you don't have an Azure subscription, create a [free account](https://azure.microsoft.com/free/?WT.mc_id=A261C142F) before you begin.

Alternatively, you can [Try Azure Cosmos DB for free](https://azure.microsoft.com/try/cosmosdb/) without an Azure subscription, free of charge and commitments. Or you can use the [Azure Cosmos DB Emulator](https://docs.microsoft.com/azure/cosmos-db/local-emulator) for this tutorial with a connection string of

Copy

mongodb://localhost:C2y6yDjf5/R+ob0N8A7Cgv30VRDJIWEHLM+4QDU5DE2nQ9nDuVTqobD4b8mGGyPMbIZnqyMsEcaGQy67XIw/Jw==@localhost:10255/admin?ssl=true

In addition to Azure CLI, you need [Node.js](https://nodejs.org/) and [Git](http://www.git-scm.com/downloads) installed locally to run npm and gitcommands.

You should have working knowledge of Node.js. This quickstart is not intended to help you with developing Node.js applications in general.

Clone the sample application

Open a git terminal window, such as git bash, and cd to a working directory.

Run the following commands to clone the sample repository. This sample repository contains the default [MEAN.js](http://meanjs.org/) application.

bashCopy

git clone https://github.com/prashanthmadi/mean

Run the application

Install the required packages and start the application.

bashCopy

cd mean

npm install

npm start

The application will try to connect to a MongoDB source and fail, go ahead and exit the application when the output returns "[MongoError: connect ECONNREFUSED 127.0.0.1:27017]".

Log in to Azure

If you are using an installed Azure CLI, log in to your Azure subscription with the [az login](https://docs.microsoft.com/en-us/cli/azure/" \l "login)command and follow the on-screen directions. You can skip this step if you're using the Azure Cloud Shell.

Azure CLICopy

az login

Add the Azure Cosmos DB module

If you are using an installed Azure CLI, check to see if the cosmosdb component is already installed by running the az command. If cosmosdb is in the list of base commands, proceed to the next command. You can skip this step if you're using the Azure Cloud Shell.

If cosmosdb is not in the list of base commands, reinstall [Azure CLI 2.0](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli).

Create a resource group

Create a [resource group](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview) with the [az group create](https://docs.microsoft.com/en-us/cli/azure/group" \l "create). An Azure resource group is a logical container into which Azure resources like web apps, databases and storage accounts are deployed and managed.

The following example creates a resource group in the West Europe region. Choose a unique name for the resource group.

If you are using Azure Cloud Shell, click **Try It**, follow the onscreen prompts to login, then copy the command into the command prompt.

Azure CLICopyTry It

az group create --name myResourceGroup --location "West Europe"

Create an Azure Cosmos DB account

Create an Azure Cosmos DB account with the [az cosmosdb create](https://docs.microsoft.com/en-us/cli/azure/cosmosdb" \l "create) command.

In the following command, please substitute your own unique Azure Cosmos DB account name where you see the <cosmosdb-name> placeholder. This unique name will be used as part of your Azure Cosmos DB endpoint (https://<cosmosdb-name>.documents.azure.com/), so the name needs to be unique across all Azure Cosmos DB accounts in Azure.

Azure CLICopyTry It

az cosmosdb create --name <cosmosdb-name> --resource-group myResourceGroup --kind MongoDB

The --kind MongoDB parameter enables MongoDB client connections.

When the Azure Cosmos DB account is created, the Azure CLI shows information similar to the following example.

Note

This example uses JSON as the Azure CLI output format, which is the default. To use another output format, see [Output formats for Azure CLI 2.0 commands](https://docs.microsoft.com/cli/azure/format-output-azure-cli).

JSONCopy

{

"databaseAccountOfferType": "Standard",

"documentEndpoint": "https://<cosmosdb-name>.documents.azure.com:443/",

"id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Document

DB/databaseAccounts/<cosmosdb-name>",

"kind": "MongoDB",

"location": "West Europe",

"name": "<cosmosdb-name>",

"readLocations": [

{

"documentEndpoint": "https://<cosmosdb-name>-westeurope.documents.azure.com:443/",

"failoverPriority": 0,

"id": "<cosmosdb-name>-westeurope",

"locationName": "West Europe",

"provisioningState": "Succeeded"

}

],

"resourceGroup": "myResourceGroup",

"type": "Microsoft.DocumentDB/databaseAccounts",

"writeLocations": [

{

"documentEndpoint": "https://<cosmosdb-name>-westeurope.documents.azure.com:443/",

"failoverPriority": 0,

"id": "<cosmosdb-name>-westeurope",

"locationName": "West Europe",

"provisioningState": "Succeeded"

}

]

}

Connect your Node.js application to the database

In this step, you connect your MEAN.js sample application to an Azure Cosmos DB database you just created, using a MongoDB connection string.

Configure the connection string in your Node.js application

In your MEAN.js repository, open config/env/local-development.js.

Replace the content of this file with the following code. Be sure to also replace the two <cosmosdb-name> placeholders with your Azure Cosmos DB account name.

JavaScriptCopy

'use strict';

module.exports = {

db: {

uri: 'mongodb://<cosmosdb-name>:<primary\_master\_key>@<cosmosdb-name>.documents.azure.com:10255/mean-dev?ssl=true&sslverifycertificate=false'

}

};

Retrieve the key

In order to connect to an Azure Cosmos DB database, you need the database key. Use the [az cosmosdb list-keys](https://docs.microsoft.com/en-us/cli/azure/cosmosdb" \l "list-keys) command to retrieve the primary key.

Azure CLICopyTry It

az cosmosdb list-keys --name <cosmosdb-name> --resource-group myResourceGroup --query "primaryMasterKey"

The Azure CLI outputs information similar to the following example.

JSONCopy

"RUayjYjixJDWG5xTqIiXjC..."

Copy the value of primaryMasterKey. Paste this over the <primary\_master\_key> in local-development.js.

Save your changes.

Run the application again.

Run npm start again.

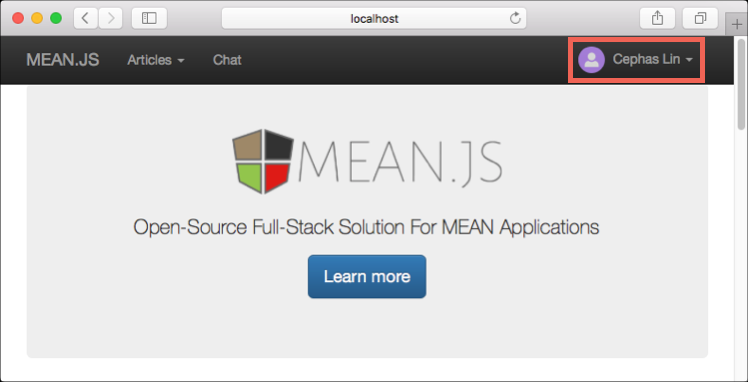
bashCopy

npm start

A console message should now tell you that the development environment is up and running.

Navigate to http://localhost:3000 in a browser. Click **Sign Up** in the top menu and try to create two dummy users.

The MEAN.js sample application stores user data in the database. If you are successful and MEAN.js automatically signs into the created user, then your Azure Cosmos DB connection is working.

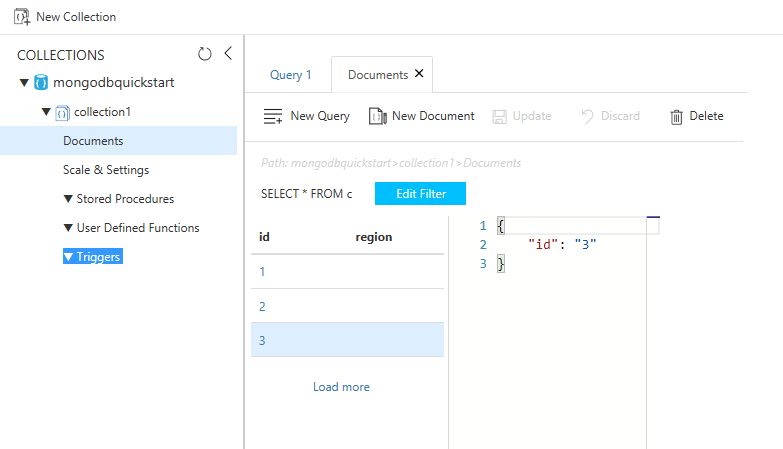


View data in Data Explorer

Data stored by an Azure Cosmos DB is available to view, query, and run business-logic on in the Azure portal.

To view, query, and work with the user data created in the previous step, login to the [Azure portal](https://portal.azure.com/)in your web browser.

In the top Search box, type Azure Cosmos DB. When your Cosmos DB account blade opens, select your Cosmos DB account. In the left navigation, click Data Explorer. Expand your collection in the Collections pane, and then you can view the documents in the collection, query the data, and even create and run stored procedures, triggers, and UDFs.



Deploy the Node.js application to Azure

In this step, you deploy your MongoDB-connected Node.js application to Azure Cosmos DB.

You may have noticed that the configuration file that you changed earlier is for the development environment (/config/env/local-development.js). When you deploy your application to App Service, it will run in the production environment by default. So now, you need to make the same change to the respective configuration file.

In your MEAN.js repository, open config/env/production.js.

In the db object, replace the value of uri as show in the following example. Be sure to replace the placeholders as before.

JavaScriptCopy

'mongodb://<cosmosdb-name>:<primary\_master\_key>@<cosmosdb-name>.documents.azure.com:10255/mean?ssl=true&sslverifycertificate=false',

Note

The ssl=true option is important because [Azure Cosmos DB requires SSL](https://docs.microsoft.com/en-us/azure/cosmos-db/connect-mongodb-account#connection-string-requirements).

In the terminal, commit all your changes into Git. You can copy both commands to run them together.

bashCopy

git add .

git commit -m "configured MongoDB connection string"

Clean up resources

If you're not going to continue to use this app, delete all resources created by this quickstart in the Azure portal with the following steps:

1. From the left-hand menu in the Azure portal, click **Resource groups** and then click the name of the resource you created.
2. On your resource group page, click **Delete**, type the name of the resource to delete in the text box, and then click **Delete**.

Next steps