# MongoDB with C# in 15 Minutes

## Purpose

This article will show how to get a C#/Windows client up and running with a [MongoDB](https://docs.mongodb.com/manual/introduction/) noSql database (or maybe just a bit longer, depending on coffee breaks).

For more details about noSql databases in general (or MongoDB in particular), look here:

<https://www.mongodb.com/nosql-explained>

## Install MongoDB on Linux

You can run MongoDB on Windows, on a Mac or in a Docker container, among many other choices. In this example, we’ll install MongoDB on an Ubuntu 18.04 Linux VM.

1. Check for and delete any obsolete/conflicting versions of MongoDB

**dpkg -l|grep -i mongo**

<= Lists all the currently installed MongoDB packages

**apt-get purge mongodb-org mongodb-org-mongos mongodb-org-server \**

**mongodb-org-shell mongodb-org-tools**

**apt-get autoremove**

<= Delete any existing MongoDB packages

**cd /var/log; rm -rf mongodb**

**cd /var/lib; rm -rf mongodb**

<= Manually delete any remaining artifacts

1. Install MongoDB from .deb

**wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | \**

**sudo apt-key add -**

<= Get certificate for this repo

**vi /etc/apt/sources.list.d/mongodb.list**

deb https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse

<= Adds repo for "apt-get"

**apt-get update**

<= Updates "apt-get" to use the new repo

**apt-get install mongodb-org**

**apt list --installed|grep -i mongo**

mongodb-org/bionic,now 4.2.0 amd64 [installed]

mongodb-org-mongos/bionic,now 4.2.0 amd64 [installed,automatic]

mongodb-org-server/bionic,now 4.2.0 amd64 [installed,automatic]

mongodb-org-shell/bionic,now 4.2.0 amd64 [installed,automatic]

mongodb-org-tools/bionic,now 4.2.0 amd64 [installed,automatic]

<= install package and verify installation

1. Start MongoDB service

**service mongod start** => OK

1. Verify basic MongoDB functionality

**ls -l /var/logo/mongodb/mongod.log**

-rw------- 1 mongodb mongodb 6832 Oct 17 13:27 mongod.log

<= Ensure that log file exists after start, check it for errors

**Mongo**

<= Start MongoDB CLI and execute a few commands

**> db.\_adminCommand( {getCmdLineOpts: 1})**

{

"argv" : [ "/usr/bin/mongod", "--config", "/etc/mongod.conf" ],

"parsed" : {

"config" : "/etc/mongod.conf",

"net" : { "bindIp" : "127.0.0.1", "port" : 27017 },

"processManagement" : { "timeZoneInfo" : "/usr/share/zoneinfo" },

"storage" : { "dbPath" : "/var/lib/mongodb",

"journal" : { "enabled" : true }

},

"systemLog" : { "destination" : "file", "logAppend" : true,

"path" : "/var/log/mongodb/mongod.log"

}

},

"ok" : 1

}

**> db.test.save( { a: 1 } )**

WriteResult({ "nInserted" : 1 })

**> db.test.find()**

{ "\_id" : ObjectId("5da8ebb6b198c5cf6ffa6122"), "a" : 1 }

<= OK: it looks like our MongoDB install is successful!

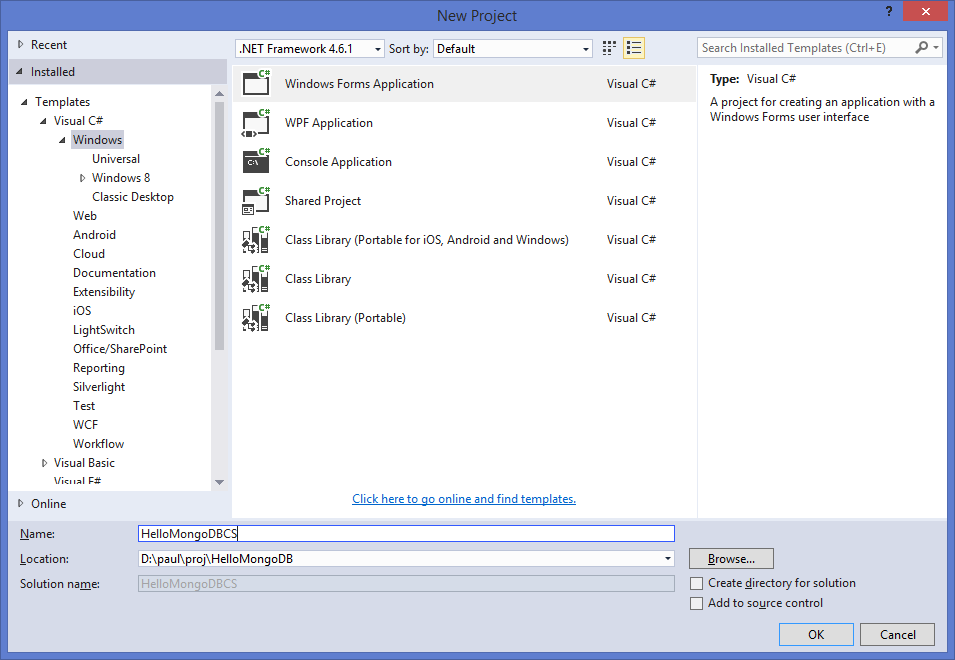
NOTE:

As you can see from the “net > bindIp” entry above, by default, MongoDB will only accept connections from “localhost” (127.0.0.1). We’ll enable remote connections (for our Windows/C# app) [below](#_Configure_MongoDB_for).

## Create C# project in Visual Studio

You can write MongoDB applications in Java, PHP, NodeJS and many other languages, including the “mongo” command line tool.

In this example, our MongoDB client will be a Windows Forms app written in C#.

* MSVS > File > New > Project > New > Visual C# > Windows > WinForms >

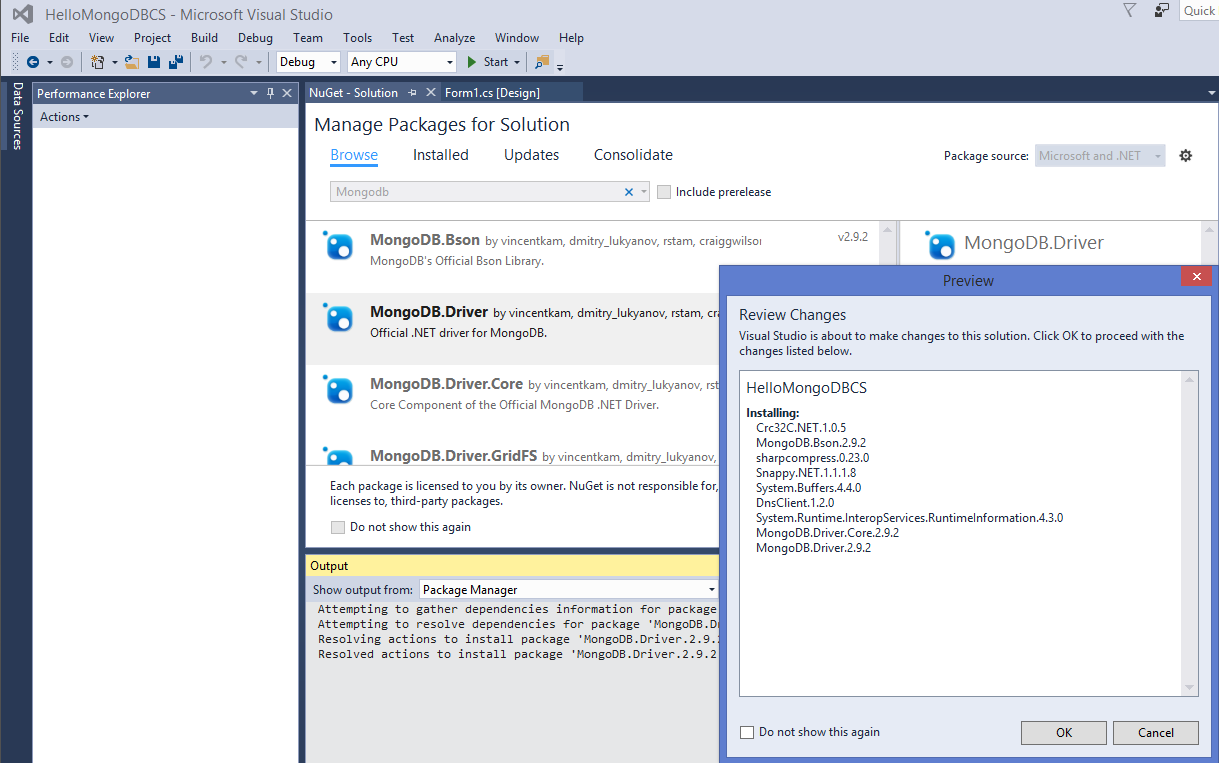
## Import MongoDB driver from NuGet

* MSCS > Tools > NuGet Package Manager >

Manage Nuget Packages for Solution > Browse >

Search for “MongoDB”

Select ”MongoDB.Driver”

"Accept" Nuget license

## Write Client code

In this simple example, we’ll have four operations:

* Connect to MongoDB
* List existing records
* Add new records
* Purge all records

We’ll create database “HelloMongoDB”, and collection “HelloMongoCollection”.

***Form1.cs:***

using MongoDB.Bson;

using MongoDB.Bson.IO;

using MongoDB.Bson.Serialization;

using MongoDB.Driver;

using System;

using System.Data;

using System.IO;

using System.Linq;

using System.Windows.Forms;

/\*\*

\* REFERENCES:\

\* https://docs.mongodb.com/manual/

\* https://www.tutorialspoint.com/mongodb/

\* https://www.c-sharpcorner.com/article/getting-started-with-mongodb-mongodb-with-c-sharp/

\* https://docs.mongodb.com/ecosystem/drivers/csharp/

\*/

namespace HelloMongoDBCS

{

public partial class Form1 : Form

{

public const string DB\_NAME = "HelloMongoDB";

public const string COLLECTION\_NAME = "HelloMongoCollection";

private bool isConnected = false;

private MongoClient mongoClient;

private IMongoDatabase db;

public Form1()

{

InitializeComponent();

}

private void btnConnect\_Click(object sender, EventArgs e)

{

try

{

if (!isConnected)

{

WriteStatus("Connecting to MongoDB on host " + edtHost.Text + ":" + edtPort.Text + "...");

// Connect to MongoDB

string connectionString = "mongodb://" + edtHost.Text + ":" + edtPort.Text;

mongoClient = new MongoClient(connectionString);

db = mongoClient.GetDatabase(DB\_NAME);

// Update UI

isConnected = true;

btnListRecords.Enabled = true;

btnAddRecords.Enabled = true;

btnPurgeRecords.Enabled = true;

btnConnect.Text = "&Disconnect";

}

else

{

WriteStatus("Disconnection from MongoDB...");

// There doesn't seem to be an explicit "Close" in the .Net API...

mongoClient = null;

isConnected = false;

btnListRecords.Enabled = false;

btnAddRecords.Enabled = false;

btnPurgeRecords.Enabled = false;

btnConnect.Text = "&Connect";

}

} catch (Exception ex)

{

WriteStatus("ERROR: " + ex);

}

}

private void btnListRecords\_Click(object sender, EventArgs e)

{

WriteStatus("Listing current records in " + COLLECTION\_NAME+ "...");

try

{

IMongoCollection<BsonDocument> collection = db.GetCollection<BsonDocument>(COLLECTION\_NAME);

var cursor = collection.Find(new BsonDocument()).ToCursor();

int i = 0;

foreach (var document in cursor.ToEnumerable())

{

using (var stringWriter = new StringWriter())

using (var jsonWriter = new JsonWriter(stringWriter))

{

BsonSerializationContext context = BsonSerializationContext.CreateRoot(jsonWriter);

collection.DocumentSerializer.Serialize(context, document);

string json = stringWriter.ToString();

WriteStatus("doc[" + i++ + ": " + json + "...");

}

}

}

catch (Exception ex)

{

WriteStatus("ERROR: " + ex);

}

}

private void btnAddRecords\_Click(object sender, EventArgs e)

{

WriteStatus("Adding records to " + COLLECTION\_NAME + "...");

try

{

var documents = Enumerable.Range(0, 5).Select(i => new BsonDocument("counter", i));

IMongoCollection<BsonDocument> collection = db.GetCollection<BsonDocument>(COLLECTION\_NAME);

collection.InsertMany(documents);

}

catch (Exception ex)

{

WriteStatus("ERROR: " + ex);

}

}

private void btnPurgeRecords\_Click(object sender, EventArgs e)

{

WriteStatus("Purging records from " + COLLECTION\_NAME + "...");

try

{

IMongoCollection<BsonDocument> collection = db.GetCollection<BsonDocument>(COLLECTION\_NAME);

collection.DeleteMany(FilterDefinition<BsonDocument>.Empty);

}

catch (Exception ex)

{

WriteStatus("ERROR: " + ex);

}

}

public void WriteStatus(string msg)

{

string prefix = DateTime.Now.ToString("MM/dd/yy HH:mm:ss");

edtStatus.AppendText(prefix + " " + msg + "\n");

}

}

}

## Configure MongoDB for remote access

By default, MongoDB will only accept connections from localhost.

Log back on to Linux and do the following:

**sudo vi /etc/mongod.conf**

# network interfaces

net:

port: 27017

bindIp: 0.0.0.0 <-- Change this to 0.0.0.0

...

#security: <-- Comment out if "authorization: enabled"

...

**ufw status**

Status: inactive

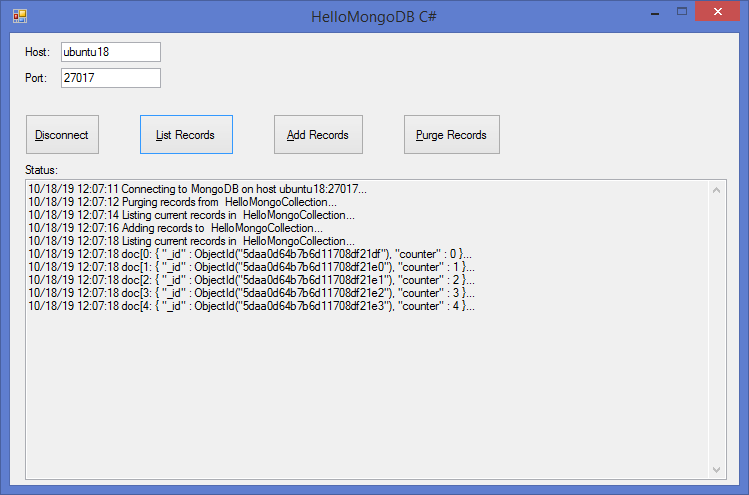
<= Firewall not active: all ports should be open on this VM...

**service mongod stop**

**service mongod start**

<= Restart Mongod

## Run and Test

Run in MSVS. Verify you can connect to MongoDB and list, add and/or purge records on the remote Linux server:

## Example code on Github