Application Development using Node JS and MongoDB

Module 7

Dr. L.M. Jenila Livingston
VIT Chennai

Introduction to Node.js

- Topics to be covered
  - Introduction to Node.js
  - Who uses Node.js?
  - What is Node.js used for?
  - What does Node.js come with?
  - Download Node.js
  - Installing Node.js
  - Selecting a Node.js IDE
  - Example Programs
  - MongoDB Connection

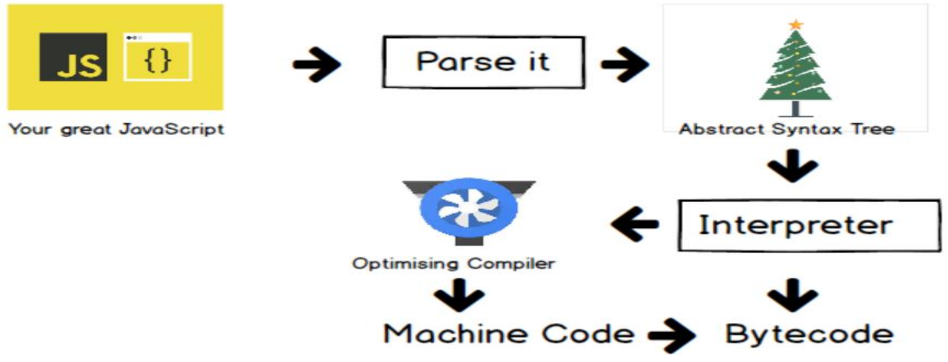
Introduction to Node.js

- Node.js is a development framework based on Google's V8 Java Script Engine for chrome web browser
- Node.js was developed in 2009 by Ryan Dahl
  - Server side environment
  - Run java script on server side

'Node's goal is to provide an easy way to build scalable network programs' - (from nodejs.org!)

Introduction to Node.js

- In ‘Node.js’ , ‘.js’ doesn’t mean that its solely written JavaScript. It is 40% JS and 60% C++.
- Asynchronous by default. So it performs faster than other frameworks.
- V8 compiles the code into machine code to be executed



Introduction to Node.js

- Node.js is a great framework
  - JavaScript end to end
    - Both server side and client side scripts
  - Event driven scalability
    - Single Threaded and High scalable
  - Extensibility
    - Simple to install
    - New modules to extend
  - Fast Implementation
    - Easy to setup Node JS and develop in it

Introduction to Node.js

- Node.js environment is
  - Clean
  - Easy to install,
  - ReadLine
    - Enables an interface to read from a data stream
  - REPL
    - Allows developers to create a command shell
  - configure and
  - Deploy

Introduction to Node.js

- Who uses Node.js?
  - Yahoo
  - LinkedIn
  - eBay
  - New York Times
  - Dow Jones
  - Microsoft

Node.js used for

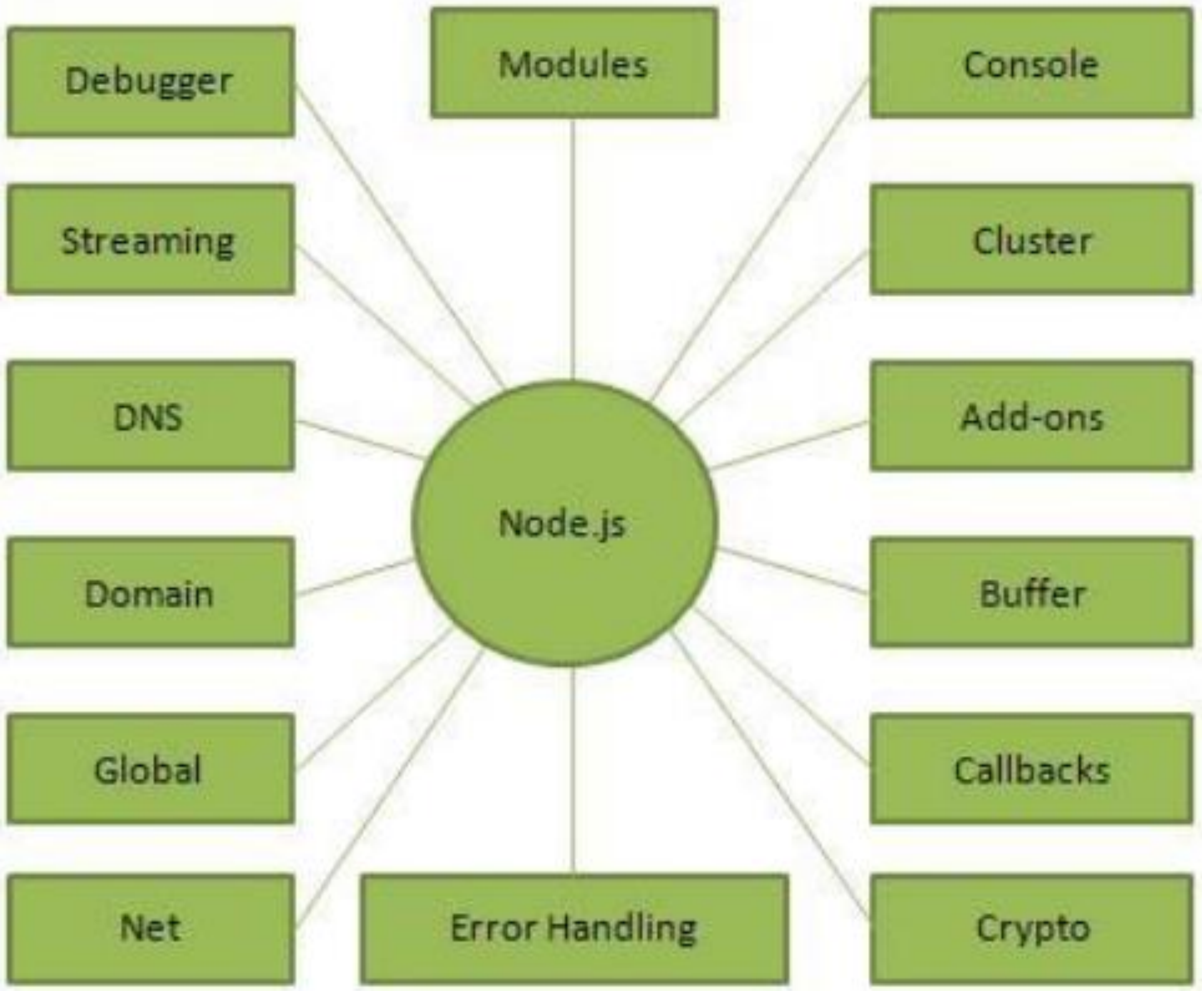
- What is Node.js used for?
  - Web services APIs such as REST (Representational State Transfer - software architectural style that defines a set of constraints to be used for creating Web services.)
  - Real-time multiplayer games
  - Backend web services such as cross-domain, server-side requests
  - Web-based applications
  - Multi client communication such as IM

Node.js/HTTP vs. Apache

- Node.js/HTTP
  - It's fast
  - It can handle tons of concurrent requests
  - It's written in JavaScript (which means you can use the same code server side and client side)

Platform	Number of request per second
PHP (via Apache)	3187,27
Static (via Apache)	2966,51
Node.js	5569,30

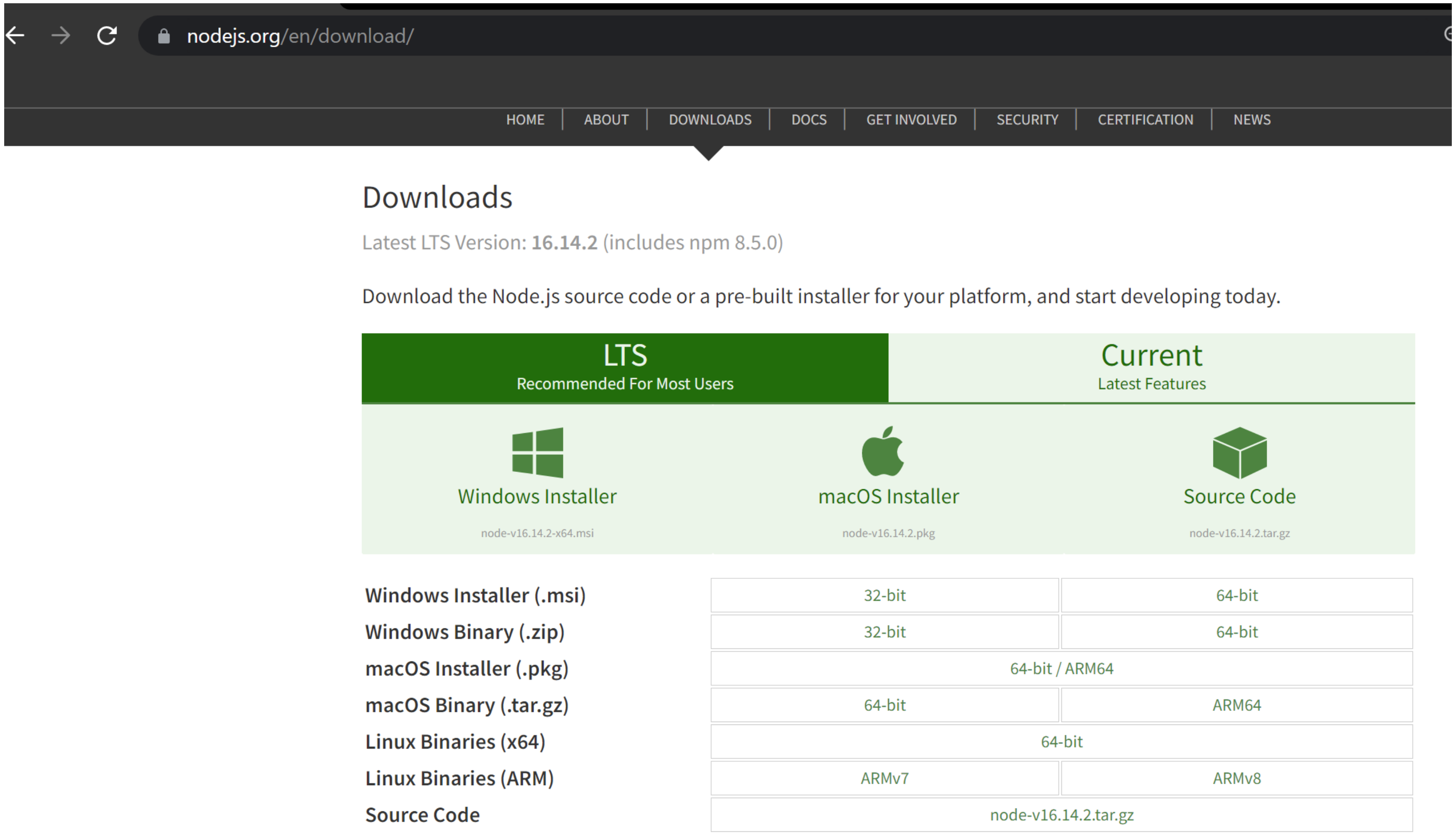
Reference: https://slideplayer.com/slide/7467003/


Debugger
Modules
Console
Streaming
Cluster
DNS
Add-ons
Node.js
Domain
Buffer
Global
Callbacks
Net
Error Handling
Crypto

Installing Node.js

- Download Node.js installer from https://nodejs.org/en/download/
- Node.js installer installs the necessary files on your PC to get Node.js up and running
- Node.js installation location

Installing Node.js



nodejs.org/en/download/

HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Downloads

Latest LTS Version: **16.14.2** (includes npm 8.5.0)

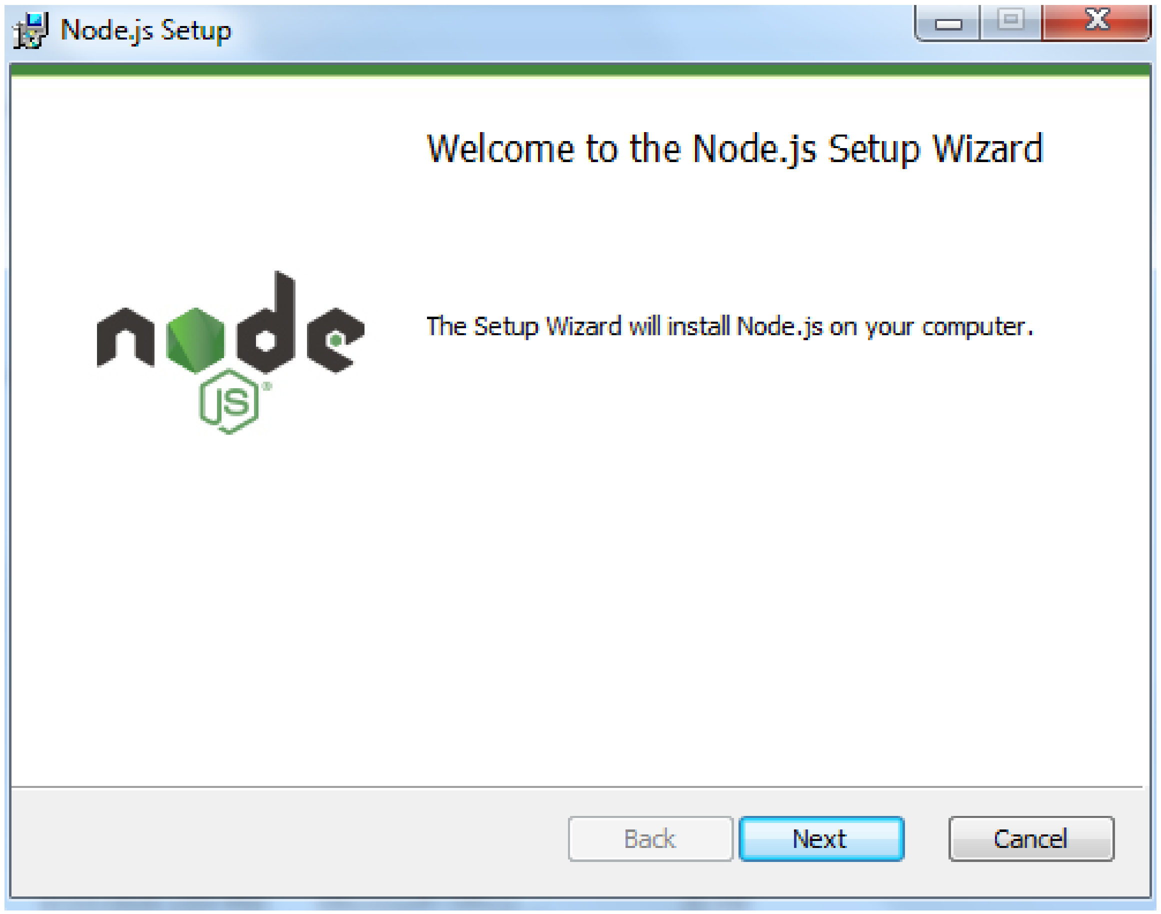
Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users	Current Latest Features

Windows Installer	macOS Installer	Source Code
node-v16.14.2-x64.msi	node-v16.14.2.pkg	node-v16.14.2.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v16.14.2.tar.gz	

Installing Node.js

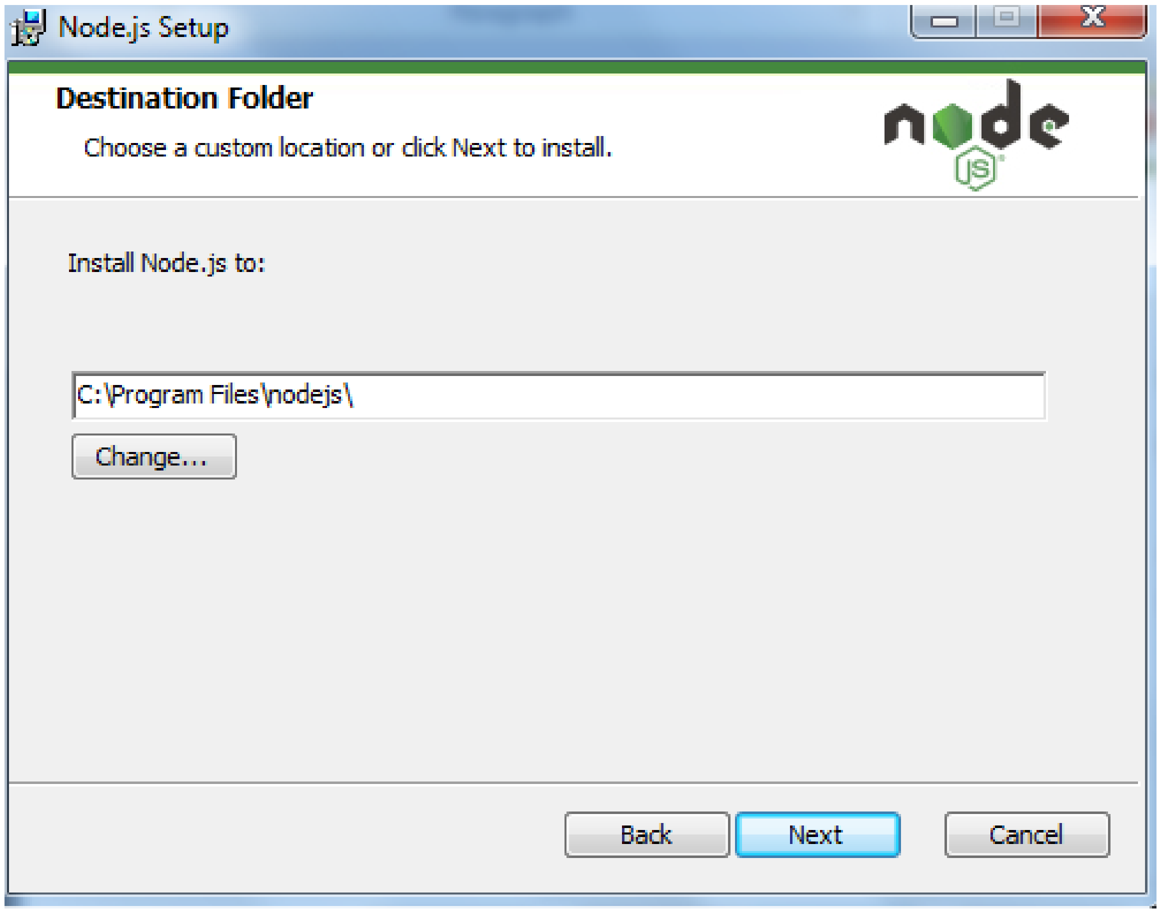


Node.js Setup

Welcome to the Node.js Setup Wizard

The Setup Wizard will install Node.js on your computer.

Installing Node.js



Node.js Setup

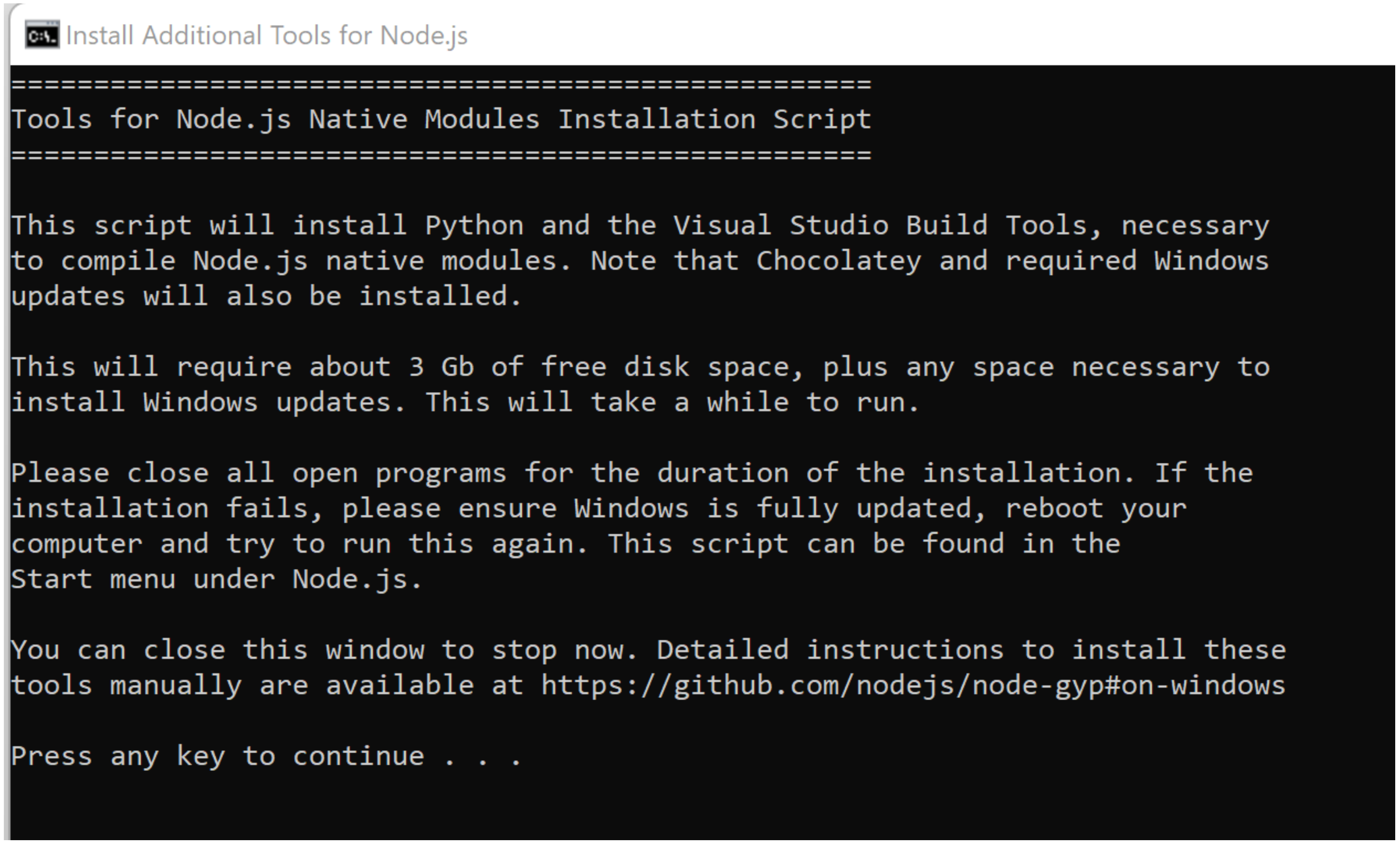
Destination Folder

Choose a custom location or click Next to install.

Install Node.js to:

C:\Program Files\nodejs\

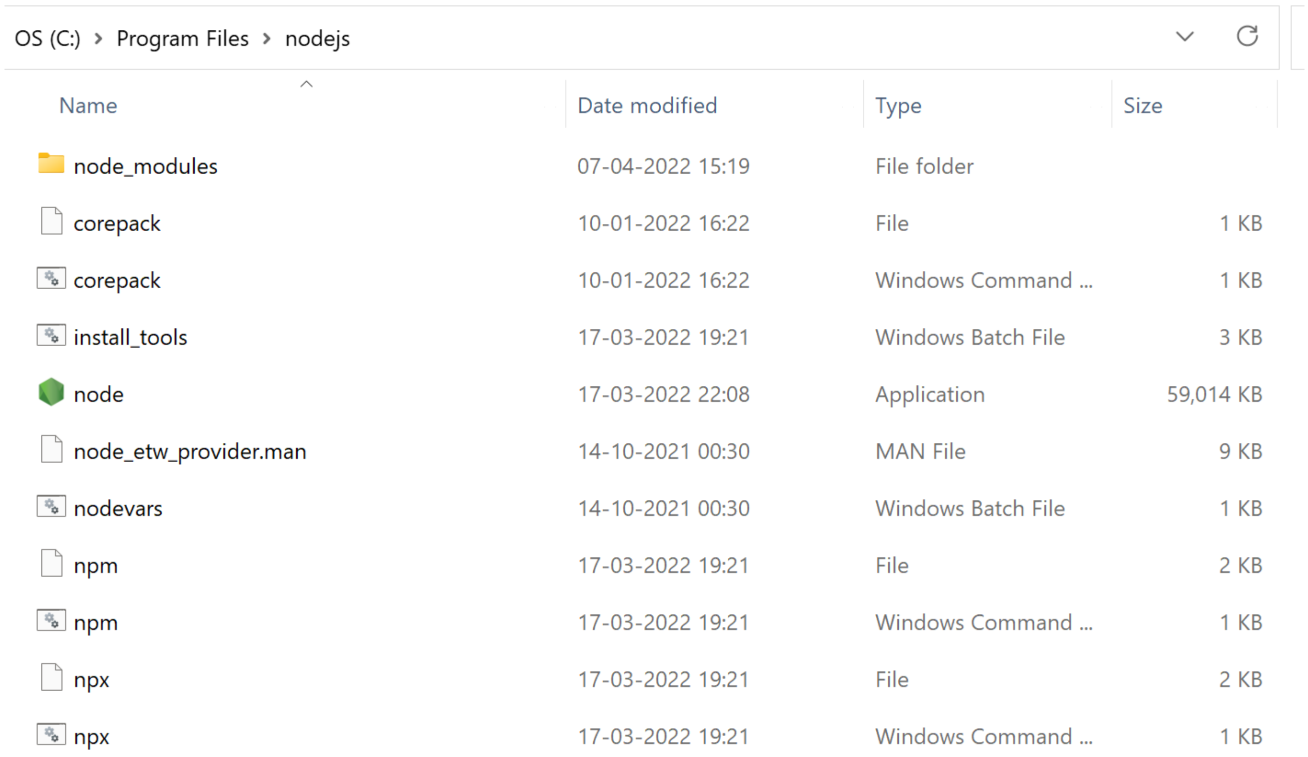
Installing Node.js



```
====================================================
Tools for Node.js Native Modules Installation Script
====================================================

This script will install Python and the Visual Studio Build Tools, necessary
to compile Node.js native modules. Note that Chocolatey and required Windows
updates will also be installed.

This will require about 3 Gb of free disk space, plus any space necessary to
install Windows updates. This will take a while to run.

Please close all open programs for the duration of the installation. If the
installation fails, please ensure Windows is fully updated, reboot your
computer and try to run this again. This script can be found in the
Start menu under Node.js.

You can close this window to stop now. Detailed instructions to install these
tools manually are available at https://github.com/nodejs/node-gyp#on-windows

Press any key to continue . . .
```

Installing Node.js

- Node – starts Node.js Java Script VM
- Npm – Node.js package manager-manages the Node.js packages
- Node_modules – consist of Node.js packages

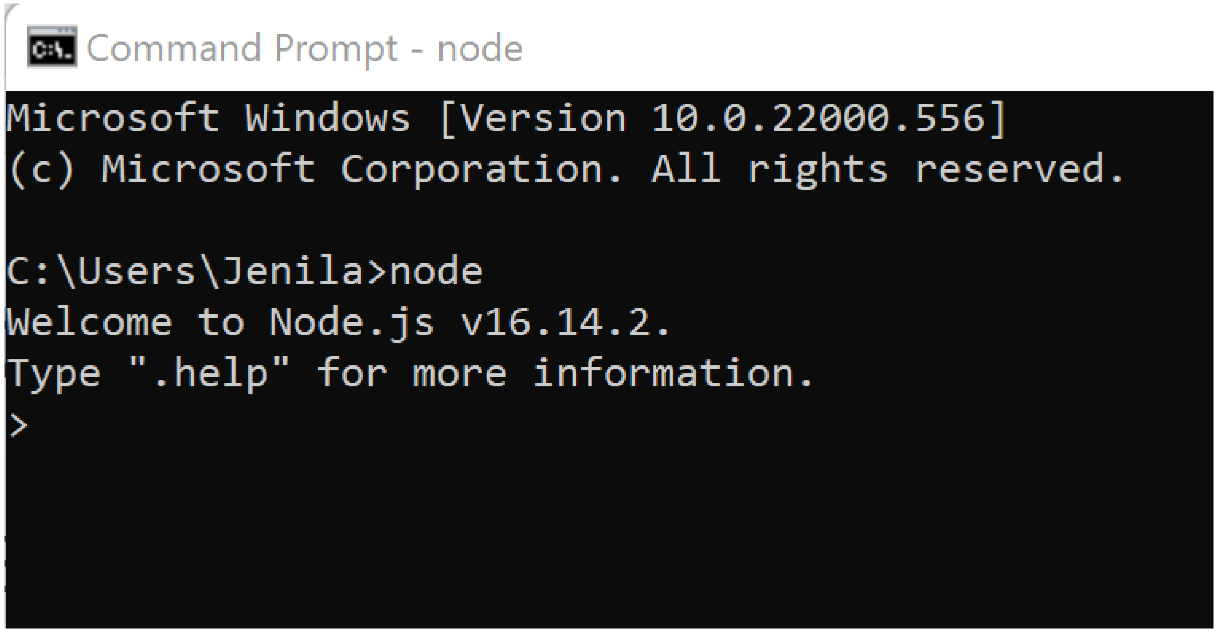


OS (C:) › Program Files › nodejs

Name	Date modified	Type	Size
node_modules	07-04-2022 15:19	File folder	
corepack	10-01-2022 16:22	File	1 KB
corepack	10-01-2022 16:22	Windows Command ...	1 KB
install_tools	17-03-2022 19:21	Windows Batch File	3 KB
node	17-03-2022 22:08	Application	59,014 KB
node_etw_provider.man	14-10-2021 00:30	MAN File	9 KB
nodevars	14-10-2021 00:30	Windows Batch File	1 KB
npm	17-03-2022 19:21	File	2 KB
npm	17-03-2022 19:21	Windows Command ...	1 KB
npx	17-03-2022 19:21	File	2 KB
npx	17-03-2022 19:21	Windows Command ...	1 KB

Installing Node.js

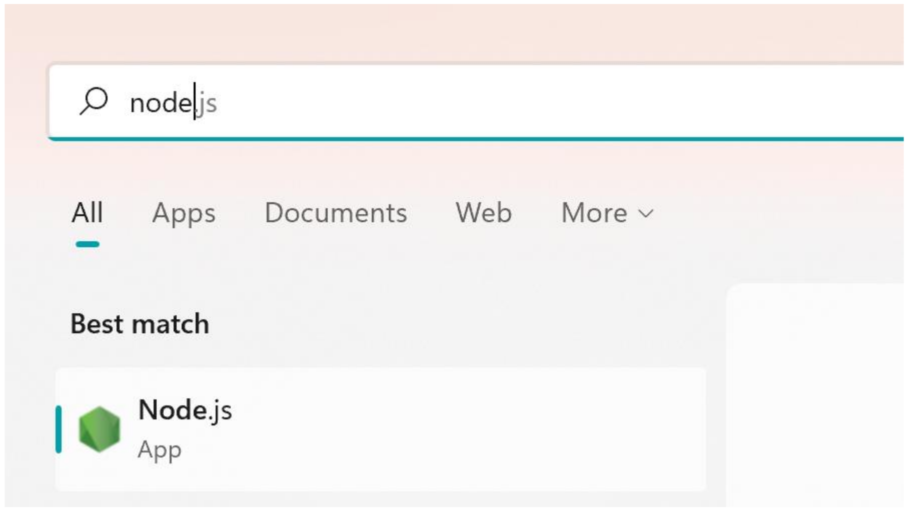
- Verify Node.js Executables
  - To verify whether Node.js is installed and working
    - Execute the command “node” in command prompt

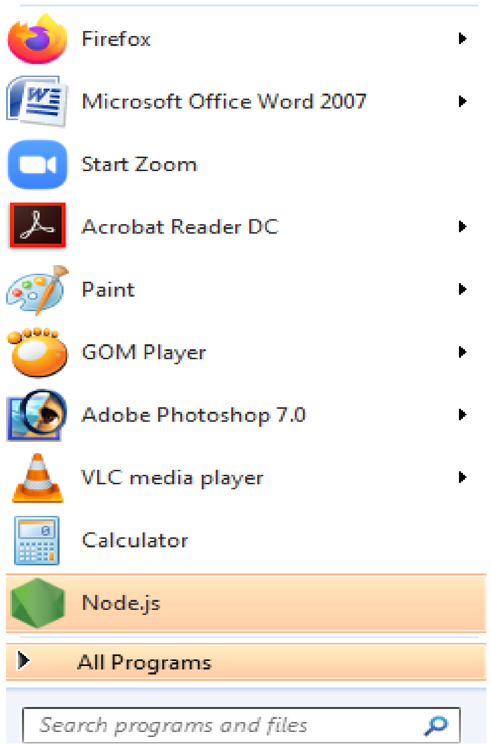


```
Microsoft Windows [Version 10.0.22000.556]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Jenila>node
Welcome to Node.js v16.14.2.
Type ".help" for more information.
>
```

Installing Node.js

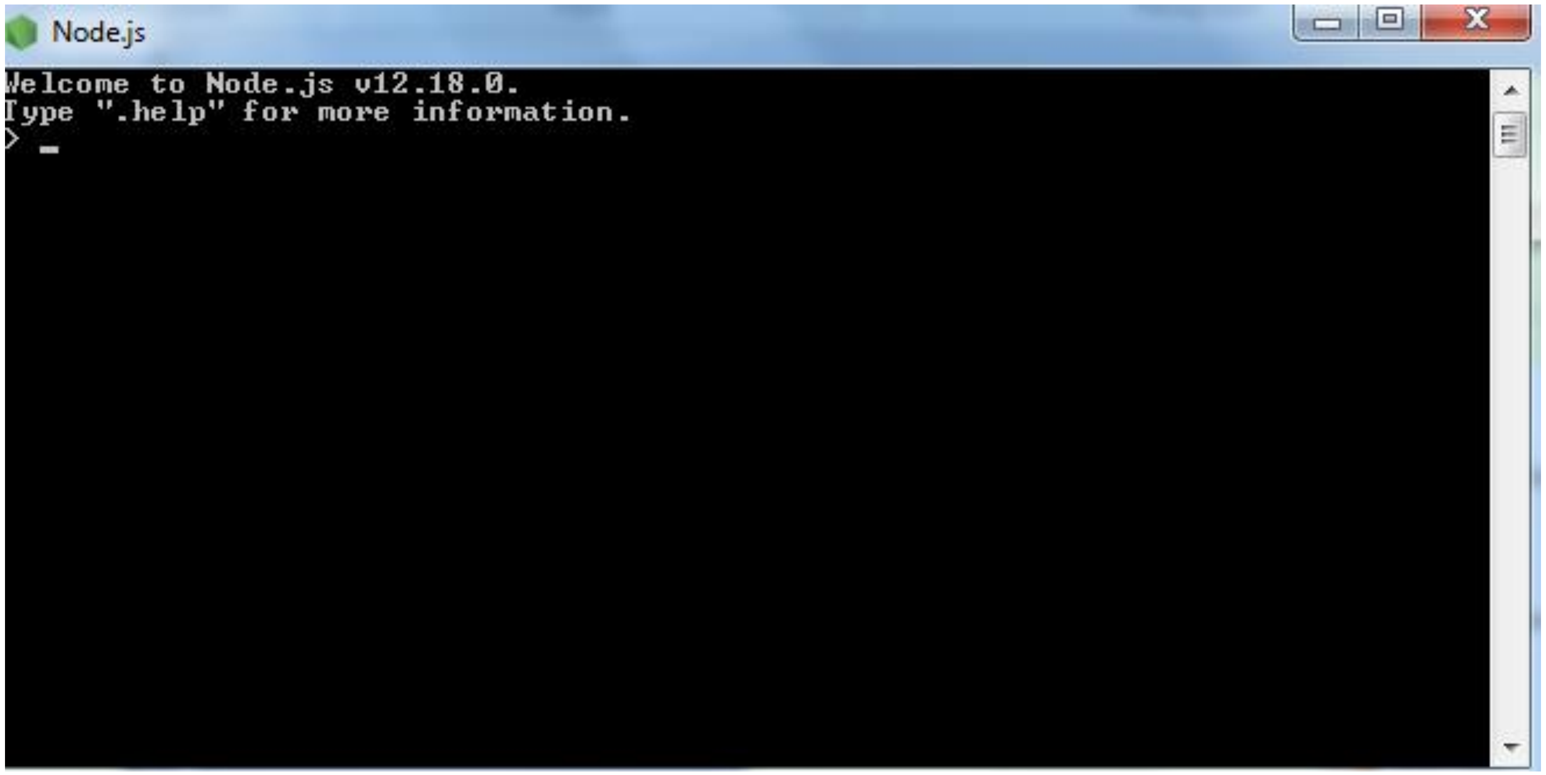
- Verify Node.js Executables
  - To verify whether Node.js is installed and working





Installing Node.js

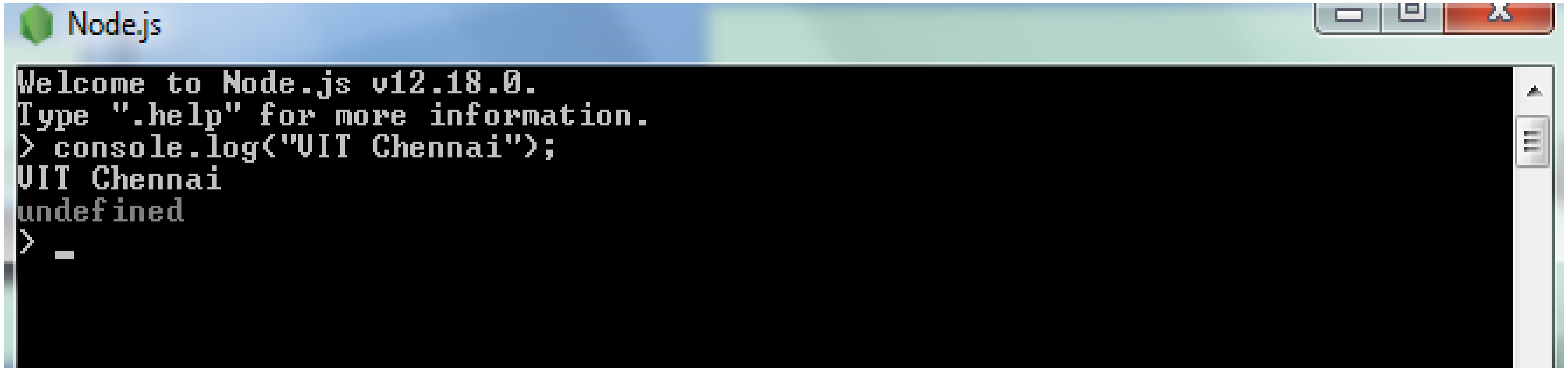
- Node.js VM



```
Welcome to Node.js v12.18.0.
Type ".help" for more information.
> _
```

Installing Node.js

- console.log("VIT Chennai")

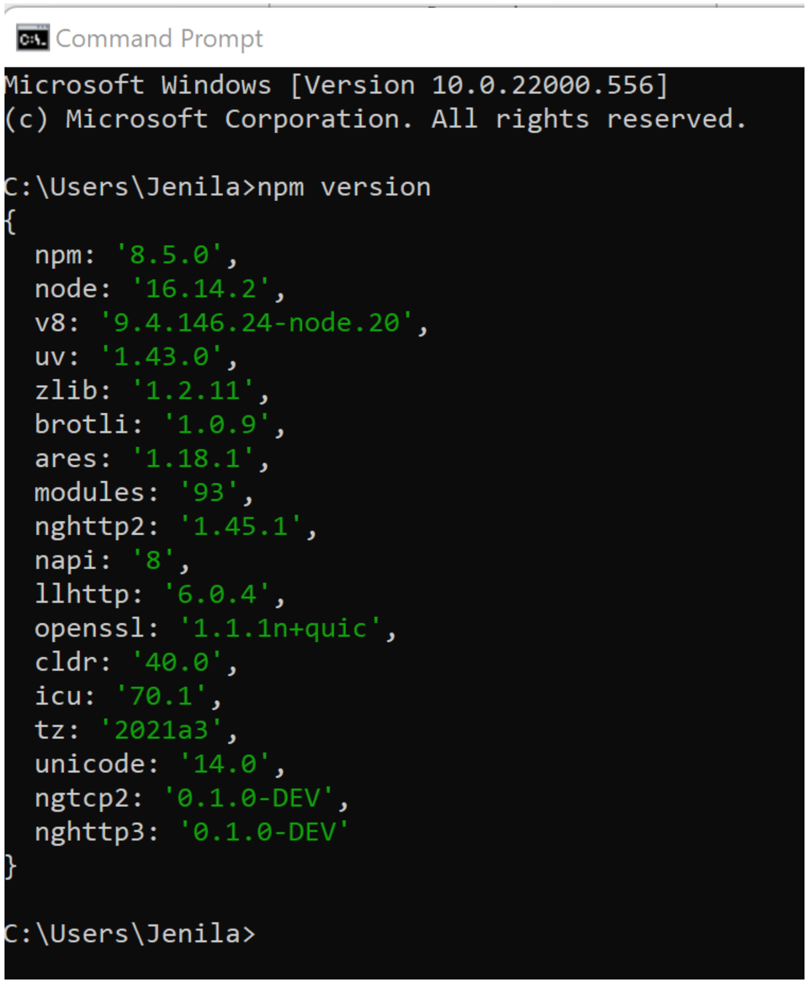


```
Welcome to Node.js v12.18.0.
Type ".help" for more information.
> console.log("VIT Chennai");
VIT Chennai
undefined
> _
```

- To exit from console window
  - Ctrl + C or Ctrl + d - Windows
  - Cmd+C - Mac

Installing Node.js

- Verify npm command is working
  - npm version



```
Microsoft Windows [Version 10.0.22000.556]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Jenila>npm version
{
  npm: '8.5.0',
  node: '16.14.2',
  v8: '9.4.146.24-node.20',
  uv: '1.43.0',
  zlib: '1.2.11',
  brotli: '1.0.9',
  ares: '1.18.1',
  modules: '93',
  nghttp2: '1.45.1',
  napi: '8',
  llhttp: '6.0.4',
  openssl: '1.1.1n+quic',
  cldr: '40.0',
  icu: '70.1',
  tz: '2021a3',
  unicode: '14.0',
  ngtcp2: '0.1.0-DEV',
  nghttp3: '0.1.0-DEV'
}

C:\Users\Jenila>
```

Installing Node.js

- Selecting a Node.js IDE
  - Eclipse
  - WebStrom
  - Text Editor
  - Code will be in
    - .js
    - .json
    - .html
    - .css

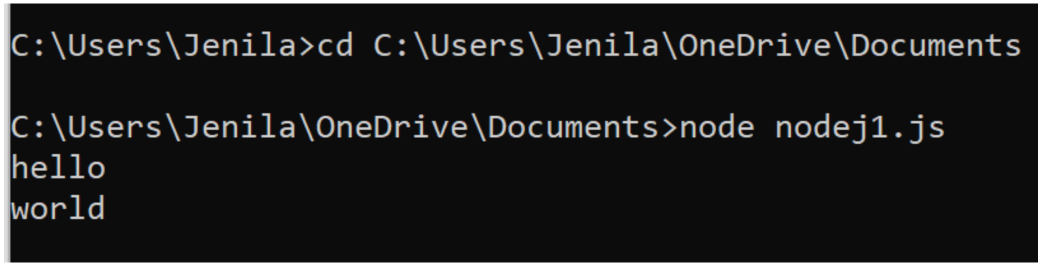
How to Run

- Type the program in Notepad
- Save with .js extension
- Go to command prompt
- Change directory
- Run by **Node pgm.js**

Example 1

//Program to print ‘hello’ first and waits for 3 seconds and then prints ‘world’

```
var util = require('util');
setTimeout(function(){console.log('world');},3000);
console.log('hello');
```



```
C:\Users\Jenila>cd C:\Users\Jenila\OneDrive\Documents

C:\Users\Jenila\OneDrive\Documents>node nodej1.js
hello
world
```

The **log**() method writes (**logs**) a message to the **console**.

Refer:
https://nodejs.org/api/timers.html

Step 1 - Import Required Module

- Use require directive to **load the http module and store** the returned **HTTP instance** into an http variable as follows:
- **var http = require("http");**

Step 2-Create Server

- Use the created http instance and **call http.createServer() method to create a server instance**
- Pass it a function with parameters **request and response.**
- **http.createServer(function (request, response) {….});**

Step 3 – Bind with Port

- Bind it at port 8081 using the **listen method** associated with the server instance.
- **server.listen(8081);**
- Testing Request & Response: create an HTTP server which listens, i.e., waits for a request over 8081 port on the local machine.

Example 2

```
var http = require("http");
var server=http.createServer(function (request, response) {
response.write('Hello\n');
response.end('Hello World\n');
});
server.listen(8081);

// Console will print the message
console.log('Server running at http://127.0.0.1:8081/ or
http://localhost:8081/');
```

response.end() is used to tell the server that the data has been loaded

readFile() method

- The **fs.readFile() method** is an inbuilt method which is used to read the file. This method read the entire file into buffer.
- **Syntax**

fs.readFile(filename, encoding, callback_function)

- To load the **fs module** we use **require()** method.
- For example: var fs = require('fs');

- **Parameters:** The method accept three parameters
- **filename:** It holds the name of the file to read
- **encoding:** It holds the encoding of file. Its default value is **'utf8'**.
- **callback_function:** It is a callback function that is called after reading of file. It takes two parameters:
  - **err:** If any error occured.
  - **data:** Contents of the file.

Example 3

```
var fs = require('fs');
fs.readFile('./sample.txt', 'utf8', function (err,data) {
if (err) {
return console.log(err); }
console.log(data);
});
```

Example 4

```
var http = require("http");
let fs = require('fs');
var server=http.createServer(function (request, response) {
// response.writeHead(200, {'Content-Type': 'text/plain'});
fs.readFile('./drink.html', null, function (error, data) {
                if (error) {
                                response.writeHead(404);
                                respone.write('Whoops! File not found!');
               }
               else {
                                response.write(data);
               }
               response.end();
               });
});
server.listen(8000);
// Console will print the message
console.log('Server running at http://127.0.0.1:8000/ or http://localhost:8000/');
```

Example 5

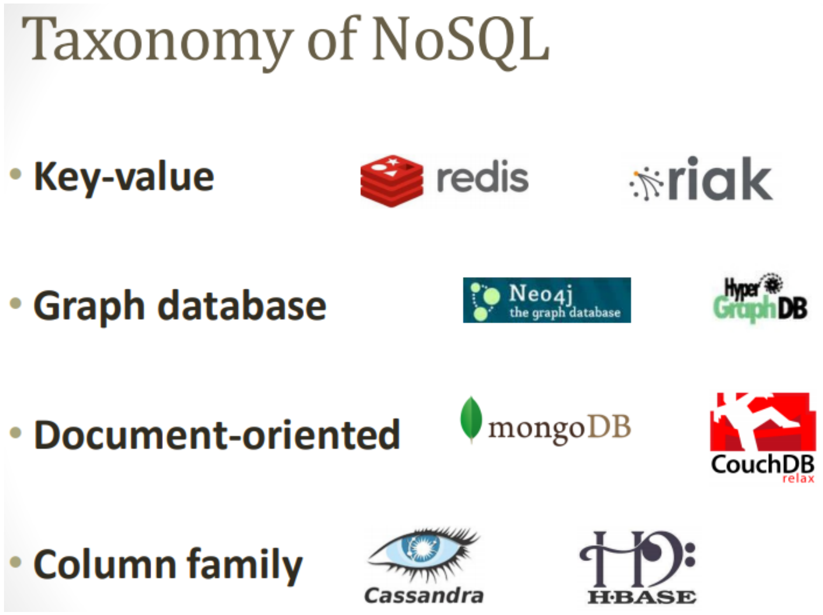
```
var http = require("http");
let fs = require('fs');
var server=http.createServer(function (request, response) {
response.writeHead(200, {'Content-Type': 'text/HTML'});
fs.readFile('./drink.html', null, function (error, data) {
      if (error) {
         response.writeHead(404);
         respone.write('Whoops! File not found!');
      }
                      else {
                      response.write(data);
               }response.end();
            });
});
server.listen(8081);
// Console will print the message
console.log('Server running at http://127.0.0.1:8081/ or http://localhost:8081/');
```

drink.html

```
<html><body>
<p>Click the button to ask for your favorite drink.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  let text;
  let favDrink = prompt("What's your favorite drink?", “Apple -Juice");
  switch(favDrink) {
    case “Apple-Juice":
      text = "Excellent choice: Apple-Juice.";
      break;
    case “Orange-Juice":
      text = "Nice choice: Orange-Juice.";
      break;
    case “Melon-Juice":
      text = "Really? Are you sure the Melon is your favorite?";
      break;
    default:
      text = "I have never heard of that one..";
  }
  document.getElementById("demo").innerHTML = text;
}
</script></body></html>
```

MongoDB

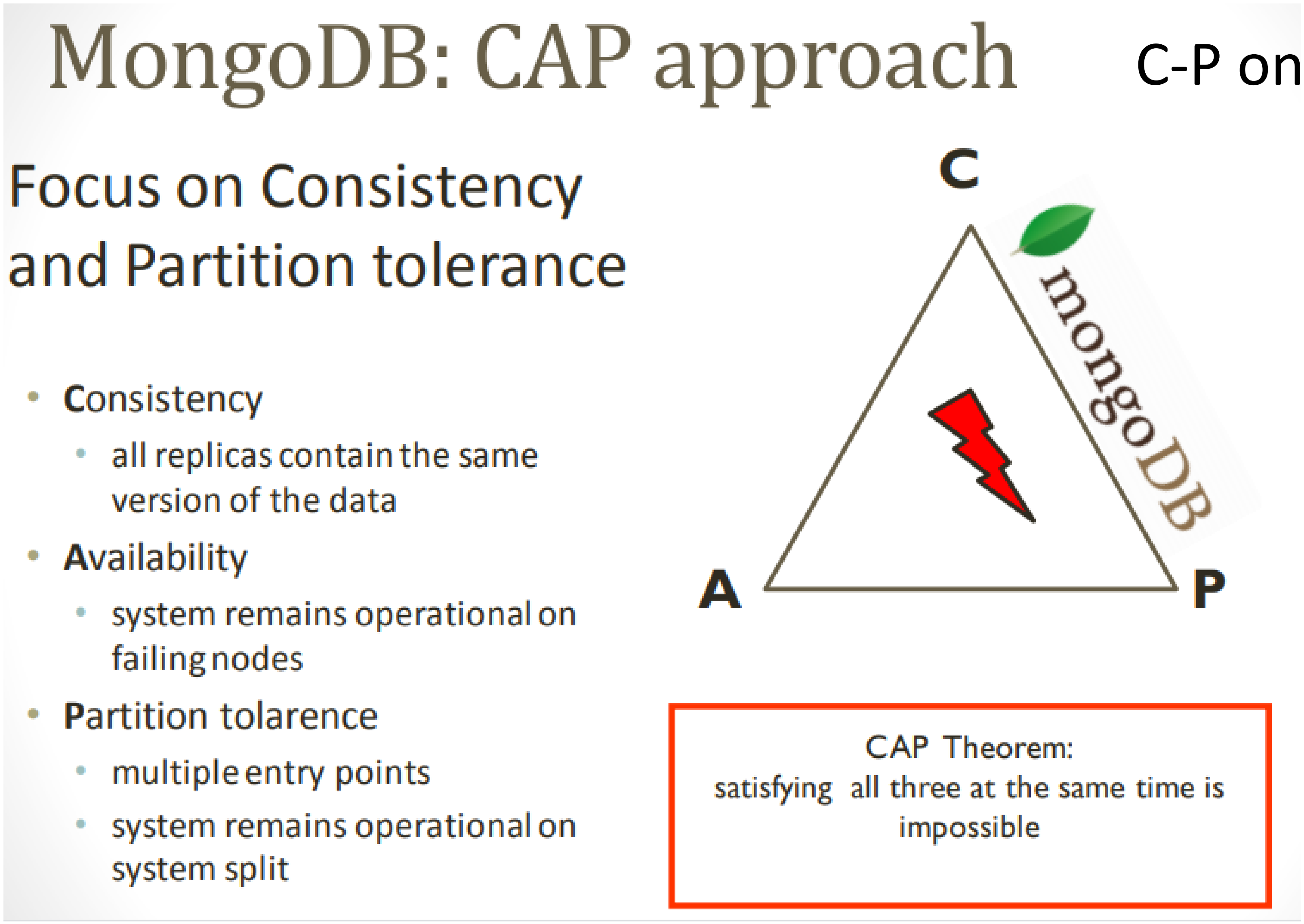
- Name comes from "Hu**mongo**us" & huge data
- Developed by 10gen
- Founded in 2007
- Written in C++, developed in 2009
- One of the most popular **NoSQL database**
  - **N**ot **O**nly **SQL**
- Document Oriented Database (max 16 MB)
- Full index for High Performance
- MongoDB stores documents or objects
- Document storage in BSON
  - Binary form of JSON
  - Binary-encoded serialization of JSON-like docs
- Each entry consists of a field name, data type, and a value
- Dynamic schema
  - No DDL



Taxonomy of NoSQL

Key-value	redis	riak
Graph database	Neo4j the graph database	HyperGraphDB
Document-oriented	mongoDB	CouchDB relax
Column family	Cassandra	HBASE

C-P on CAP



MongoDB: CAP approach

Focus on Consistency and Partition tolerance

- Consistency
  - all replicas contain the same version of the data
- Availability
  - system remains operational on failing nodes
- Partition tolarence
  - multiple entry points
  - system remains operational on system split

C
A
P
mongoDB

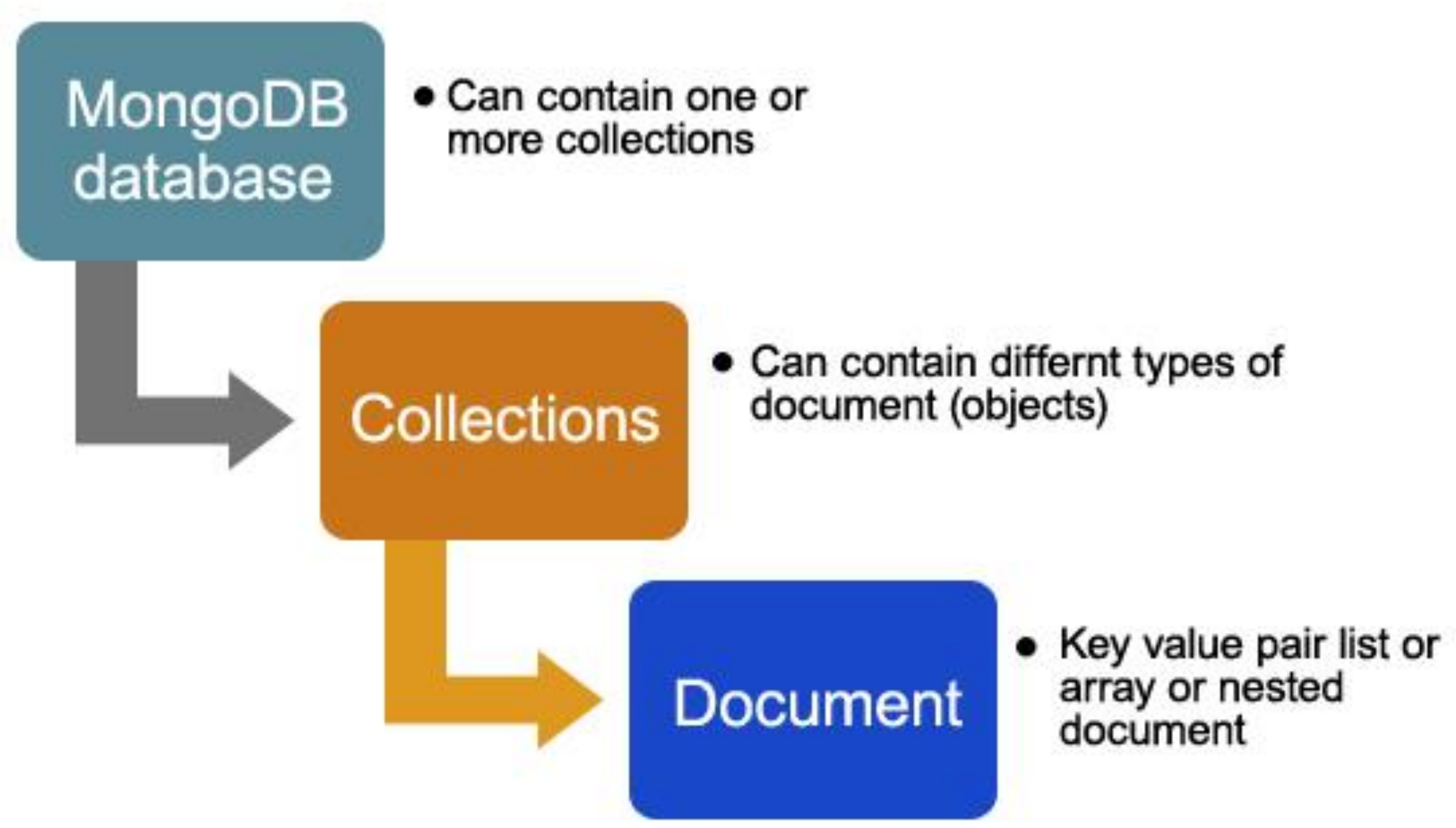
CAP Theorem:
satisfying all three at the same time is impossible

Source: https://www.ccs.neu.edu/home/kathleen/classes/cs3200/20-NoSQLMongoDB.pdf

Integration with Others

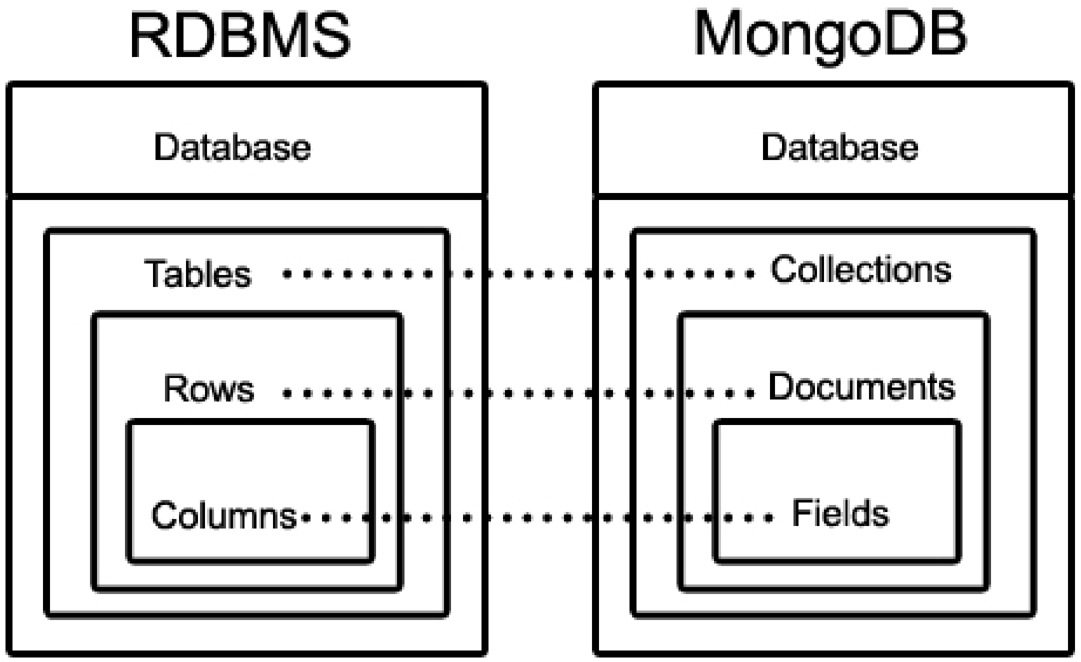
THE C PROGRAMMING LANGUAGE
C++ BJARNE STROUSTRUP
Expert F#
C#
php
python
GO
nodeJS
ERLANG
CF
Scala
Java
HASKELL
http://www.mongodb.org/display/DOCS/Drivers

MongoDB: Hierarchical Objects


MongoDB database
• Can contain one or more collections
Collections
• Can contain differnt types of document (objects)
Document
• Key value pair list or array or nested document

Source: https://www.educba.com/what-is-mongodb/

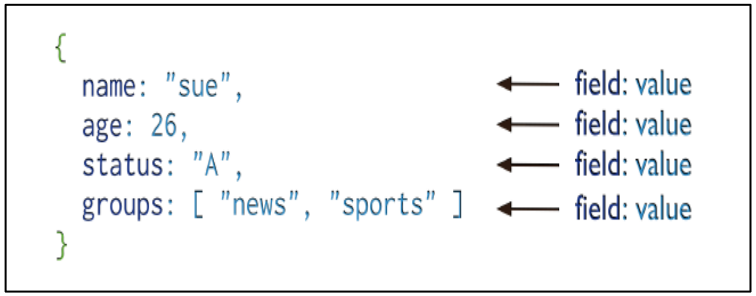
Mapping RDBMS to MongoDB



Source: https://www.educba.com/what-is-mongodb/

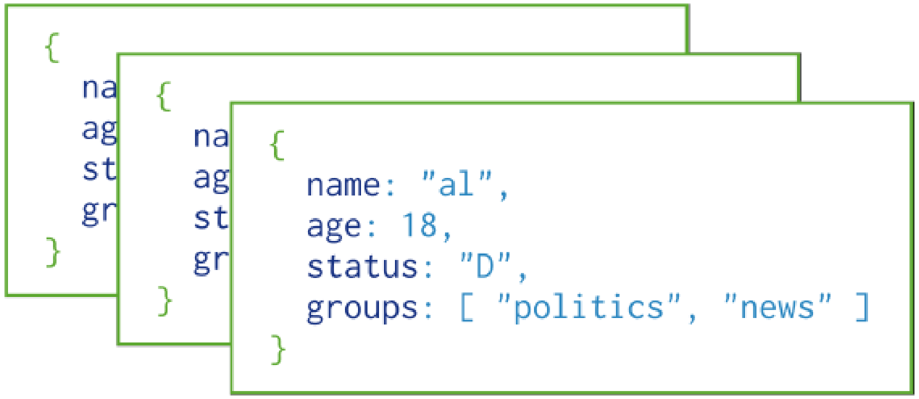
MongoDB Model

One ***document*** (e.g., one tuple in RDBMS)



```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

One ***Collection*** (e.g., one Table in RDBMS)



```
{
  name: "al",
  age: 18,
  status: "D",
  groups: [ "politics", "news" ]
}
```

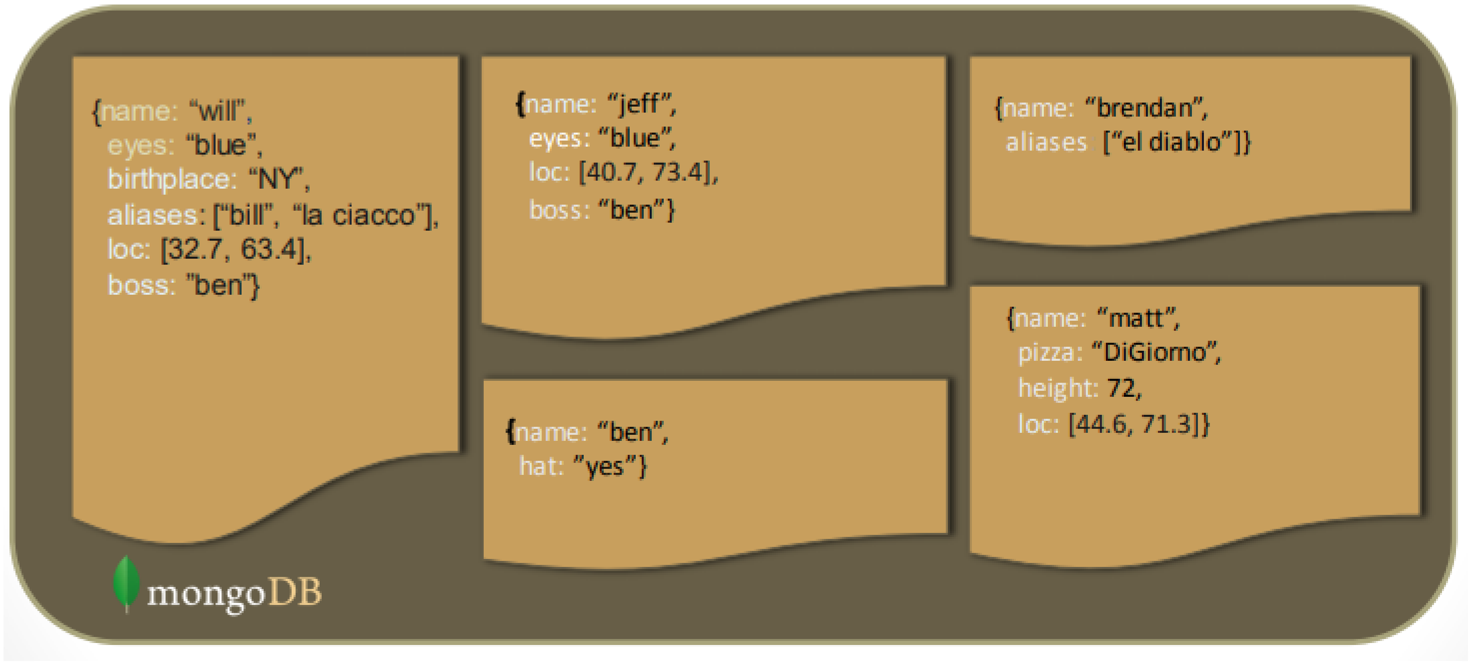
Collection

- **Collection** is a group of similar documents
- Within a collection, each document must have a unique Id

Unlike RDBMS:
No Integrity Constraints in MongoDB

Schema Free

- MongoDB does not need any pre-defined data schema
- Every document in a collection could have different data



```
{name: "will",
 eyes: "blue",
 birthplace: "NY",
 aliases: ["bill", "la ciacco"],
 loc: [32.7, 63.4],
 boss: "ben"}
```

```
{name: "jeff",
 eyes: "blue",
 loc: [40.7, 73.4],
 boss: "ben"}
```

```
{name: "brendan",
 aliases ["el diablo"]}
```

```
{name: "ben",
 hat: "yes"}
```

```
{name: "matt",
 pizza: "DiGiorno",
 height: 72,
 loc: [44.6, 71.3]}
```

JSON format

- Data is in name / value pairs
- A name(key)/value pair consists of a field name followed by a colon, followed by a value:
  - Example: “name”**:** “Leni”
- Data is separated by commas
  - Example: “name”: “Leni”**,** Address : “ABABAB”
- Curly braces hold objects
  - Example: **{**“name”: “Leni”, Address : “ABABAB” **}**
- An array is stored in brackets []
  - Example

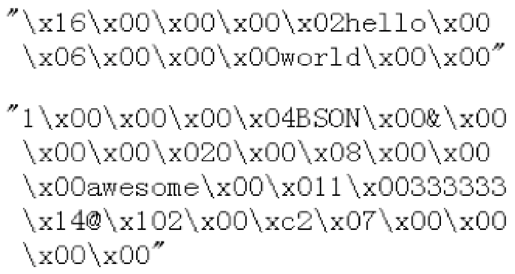
  [{“name”: “Leni”, Address : “ABABAB” },

  {“name”: “Yoda”, affiliation: “rebels”} **]**

Another Example

```
{ author: 'joe',
  created : new Date('03/28/2009'),
  title : 'Yet another blog post',
  text : 'Here is the text...',
  tags : [ 'example', 'joe' ],
  comments : [
                    { author: 'jim',
                      comment: 'I disagree'
                    },
                    { author: 'nancy',
                      comment: 'Good post'
                    }
  ]
}
```

Remember it is stored in binary formats (BSON)



```
"\x16\x00\x00\x00\x02hello\x00
 \x06\x00\x00\x00world\x00\x00"

"1\x00\x00\x00\x04BSON\x00&\x00
 \x00\x00\x020\x00\x08\x00\x00
 \x00awesome\x00\x011\x00333333
 \x14@\x102\x00\xc2\x07\x00\x00
 \x00\x00"
```

BSON Types

Type	**Number**
Double	1
String	2
Object	3
Array	4
Binary data	5
Object id	7
Boolean	8
Date	9
Null	10
Regular Expression	11
JavaScript	13
Symbol	14
JavaScript (with scope)	15
32-bit integer	16
Timestamp	17
64-bit integer	18
Min key	255
Max key	127

http://docs.mongodb.org/manual/reference/bson-types/

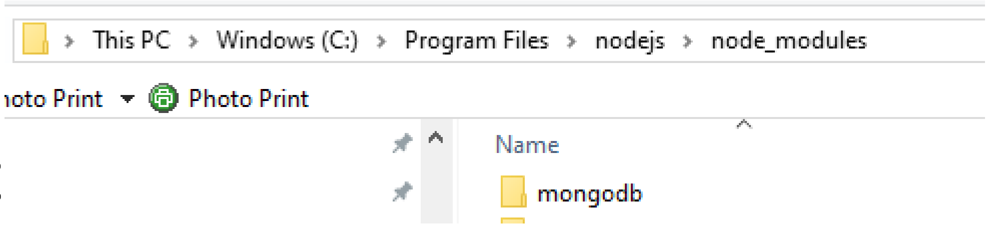
The _id Field

- By default, each document contains an _id field. This field has a number of special characteristics:
  - Value serves as primary key for collection.
  - Value is unique, immutable, and may be any non-array type.
  - Default data type is ObjectId, which is “small, likely unique, fast to generate, and ordered.” Sorting on an ObjectId value is roughly equivalent to sorting on creation time.

http://docs.mongodb.org/manual/reference/bson-types/

Part 1: Download mongodb Compass

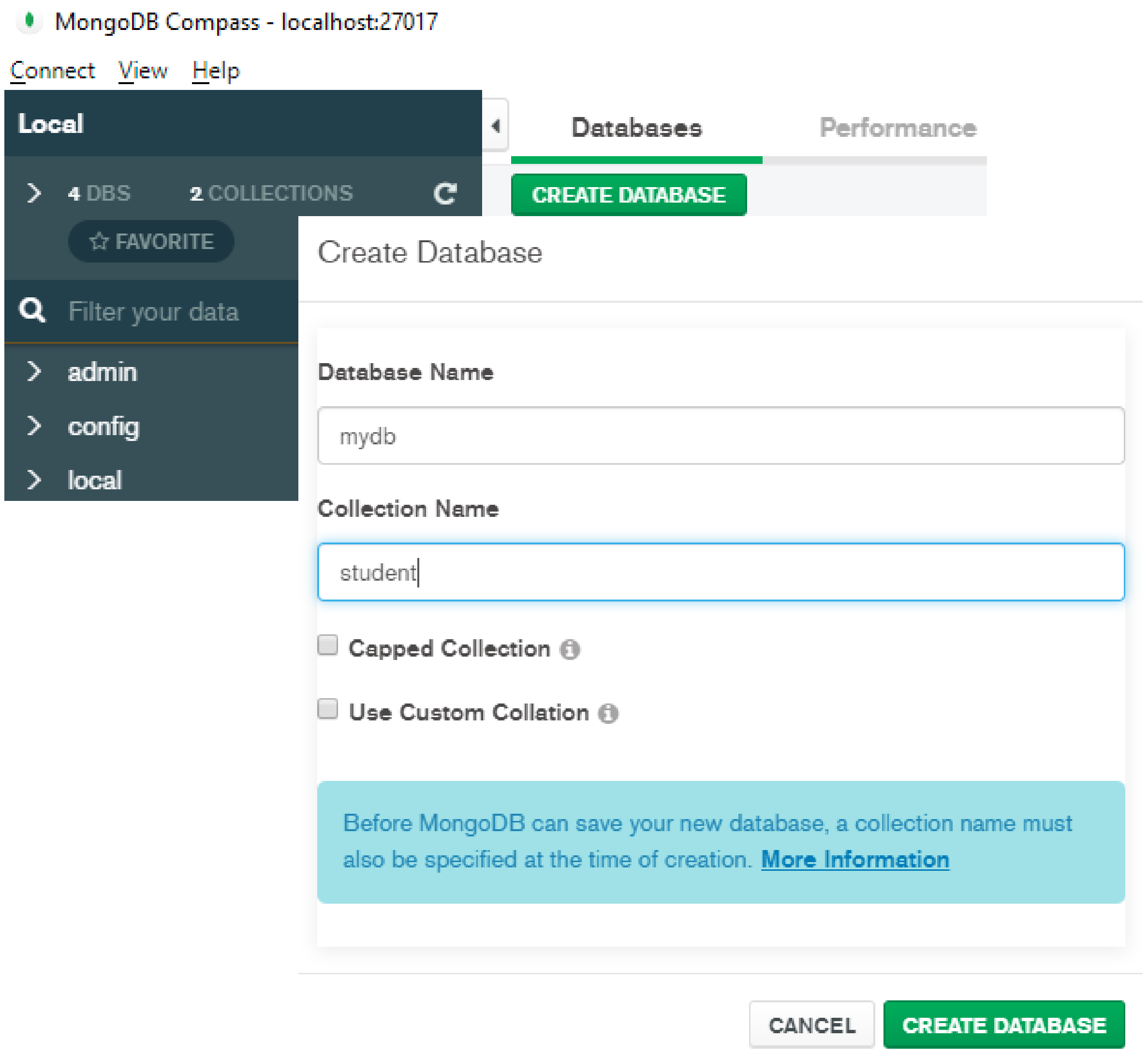
- Download and install from
- https://www.mongodb.com/try/download/community
- Connecting to NodeJS



This PC > Windows (C:) > Program Files > nodejs > node_modules

- C
- Type **npm install mongodb**
  - **Node** Package Manager
  - it is an online repository for the publishing of open-source **Node**. js projects;
  - it is a command-line utility for interacting with said repository that aids in package installation

Create database & Collection



MongoDB Compass - localhost:27017

Create Database

Database Name

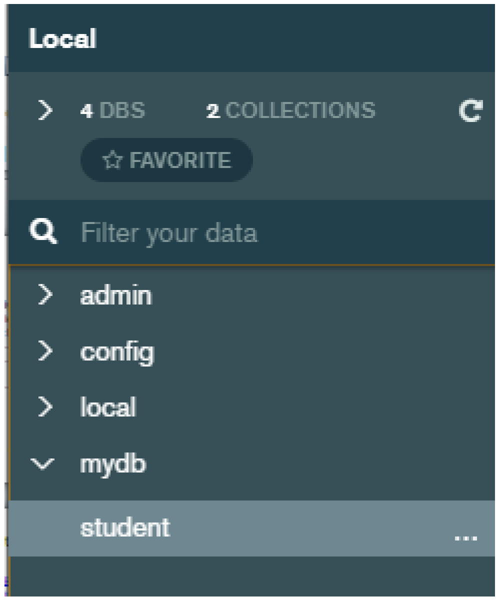
mydb

Collection Name

student

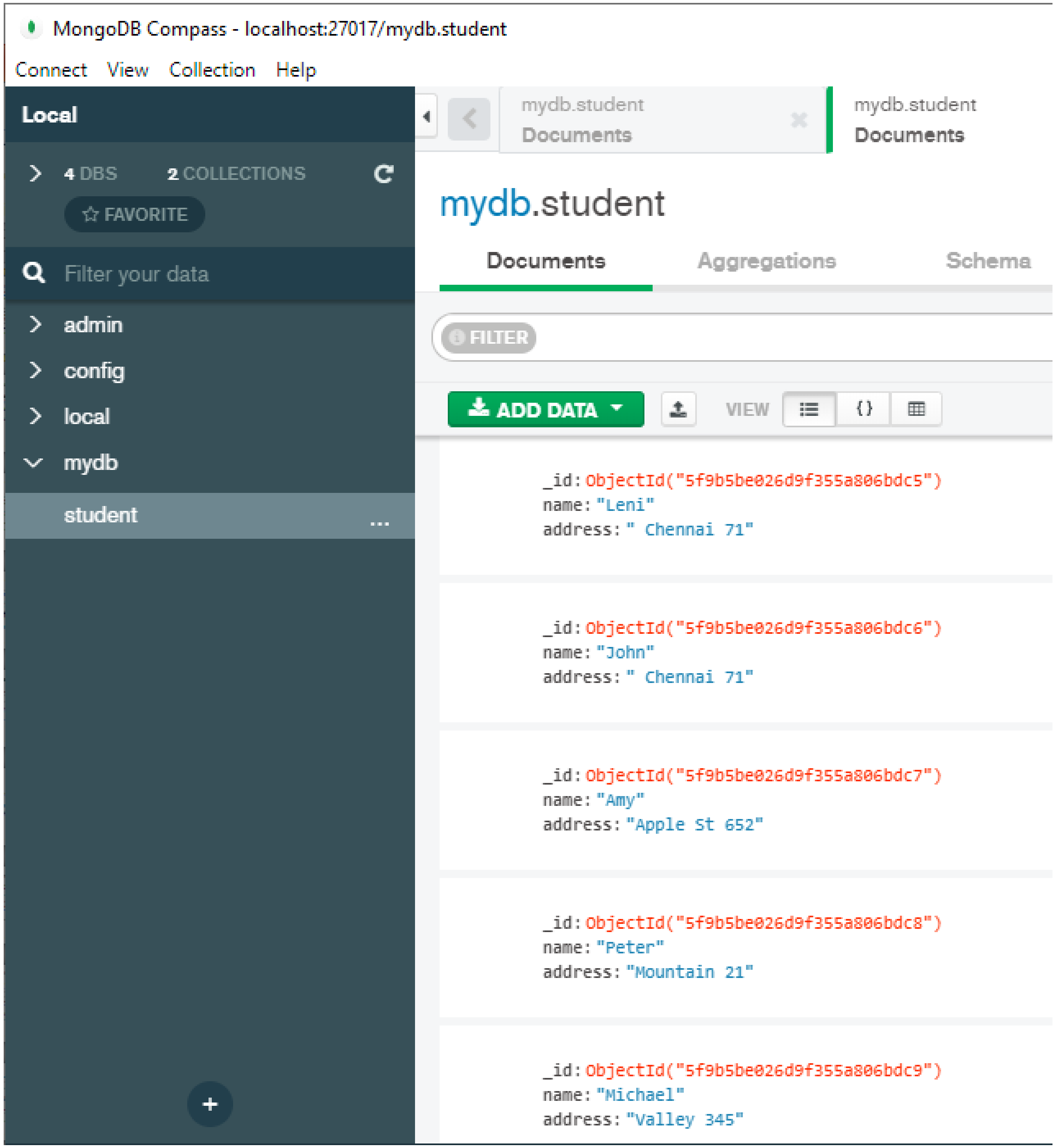
- [ ] Capped Collection
- [ ] Use Custom Collation

Before MongoDB can save your new database, a collection name must also be specified at the time of creation. More Information



Example – Inserting Multiple Records

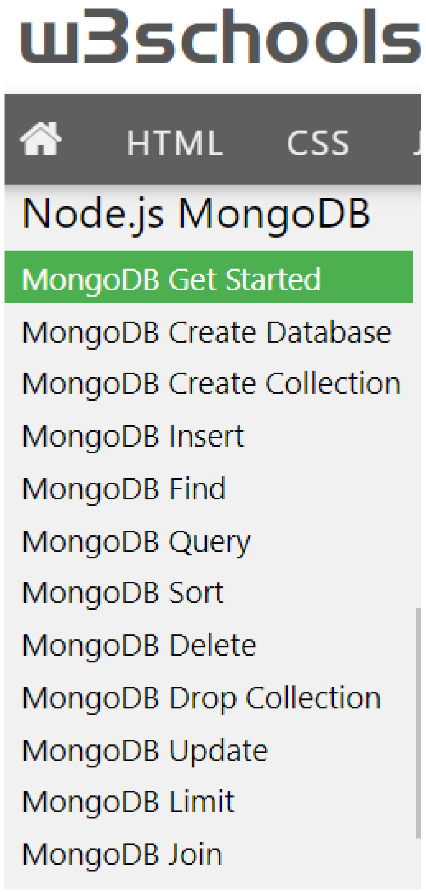
```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myobj = [
    { name: 'Leni', address: ' Chennai 71'},
    { name: 'John', address: ' Chennai 71'},
    { name: 'Amy', address: 'Apple St 652'},
    { name: 'Peter', address: 'Mountain 21'},
    { name: 'Michael', address: 'Valley 345'},
    { name: 'Sandy', address: 'Ocean St 2'},
     ];
  dbo.collection("student").insertMany(myobj, function(err, res) {
    if (err) throw err;
    console.log("Number of documents inserted: " + res.insertedCount);
    db.close();
  });
});
```



```
_id: ObjectId("5f9b5be026d9f355a806bdc5")
name: "Leni"
address: " Chennai 71"

_id: ObjectId("5f9b5be026d9f355a806bdc6")
name: "John"
address: " Chennai 71"

_id: ObjectId("5f9b5be026d9f355a806bdc7")
name: "Amy"
address: "Apple St 652"

_id: ObjectId("5f9b5be026d9f355a806bdc8")
name: "Peter"
address: "Mountain 21"

_id: ObjectId("5f9b5be026d9f355a806bdc9")
name: "Michael"
address: "Valley 345"
```

Program

- Try programs from
- https://www.w3schools.com/nodejs/nodejs_mongodb.asp



Database Creation

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

CRUD

- **C**reate
  - db.collection.insert(<document>)
  - db.collection.save(<document>)
  - db.collection.update(<query>, <update>, { upsert: true })
- **R**ead
  - db.collection.find(<query>, <projection>)
  - db.collection.findOne(<query>, <projection>)
- **U**pdate
  - db.collection.update(<query>, <update>, <options>)
- **D**elete
  - db.collection.remove(<query>, <justOne>)

CRUD

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
//open an existing database
  var dbo = db.db("mydb");

 //type your CRUD code here
});
```

Create Collection

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
//open existing database
  var dbo = db.db("mydb");

  dbo.createCollection("student", function(err, res) {
    if (err) throw err;
    console.log("Collection created!");
    db.close();
  });
});
```

Insert a Single record

```
var myobj = [
   { name: 'John', address: Chennai 71'},
];
 collection.insertOne(myobj);
 console.log('Object Inserted')
```

Example: Insert

-
  ```
  var MongoClient = require('mongodb').MongoClient;
  var url = "mongodb://localhost:27017/";

  MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("mydb");
    var myobj = {name: 'John', address: Chennai 71' };
    dbo.collection("student").insertOne(myobj, function(err,
  res) {
      if (err) throw err;
      console.log("1 document inserted");
      db.close();
    });
  });
  ```

Insert Multiple records

```
var myobj = [
  { name: ‘Leni', address: ' Chennai 71'},
  { name: 'John', address: ' Chennai 71'},
  { name: 'Amy', address: 'Apple St 652'},
  { name: ‘Peter', address: 'Mountain 21'},
  { name: 'Michael', address: 'Valley 345'},
  { name: 'Sandy', address: 'Ocean St 2'},
];
 dbo.collection(“student").insertMany(myobj);
 console.log('Objects Inserted')
 client.close();
```

Find First Record

//Find the first document in the students collection:

```
dbo.collection(“student").findOne({}, function(err, result) {
if (err) throw err;
console.log(result.name);
});
```

Find All and Display Records

//Find the All document in the students collection:

```
dbo.collection(“student"). find({}).toArray(function(err, result) {
          if (err) throw err;
  console.log(result.name);
 });
```

Find and Limit the Result

//Find and limit five records

```
  collection.find().limit(5).toArray(function(err, result)
   if (err) throw err;
   console.log(result);
  );
```

Find and project few attributes

//Return the fields "name" and "address" of all documents:

// 1 or true to include the field, 0 or false to exclude the field.

```
collection.find({}, { projection: { _id: 0, name: 1, address: 1 }
}).toArray(function(err, result) {
    if (err) throw err;
   console.log(result.name);
  });
```

Filter/ Find the specific Record

```
//Find the specific record from the students collection:
   var query = { address: ”Chennai 71” };
  collection.find(query).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
});
```

Sort Records

```
{ name: 1 } // ascending
{ name: -1 } // descending
```

```
//Sort Records in Ascending order
 var mysort = { name: 1 };
 collection.find().sort(mysort).toArray(function(err, result) {
   if (err) throw err;
   console.log(result);
});
```

Note:

db.collection.find().sort({age:-1}).limit(1) // **for MAX**

db.collection.find().sort({age:+1}).limit(1) // **for MIN**

Aggregate

Count, Sum

db.collection_name.aggregate(aggregate_operation)

$sum	adds up the definite values of every document of a collection.
$avg	computes the average values of every document of a collection.
$min	finds and returns the minimum of all values from within a collection.
$max	finds and returns the maximum of all values from within a collection.

```
db.programmers.aggregate([{$group : {_id: "$type", TotalRecords: {$sum : 1}}}])
```

If the query finds more than one record, only the first occurrence is updated.

Update One record

```
//Update one record
  var myquery = { address: “Chennai 71" };
  var newvalues = { $set: {name: “Jaff", address: “Chennai 88" } };
  collection.updateOne(myquery, newvalues, function(err, res) {
    if (err) throw err;
    console.log(result);
});
```

Update Multiple records

Update all documents where the name starts with the letter “L":

```
//Update Multiple records
 var myquery = { address: /^L/ };      // Ends with L$
 var newvalues = { $set: {name: “Jaff", address: “Chennai 88" } };
 collection.updateMany(myquery, newvalues, function(err, res) {
   if (err) throw err;
   console.log(result);
});
```

Delete One Record

If the query finds more than one document, only the first occurrence is deleted.

```
//Delete one record
   var myquery = { address: 'Mountain 21' };
  collection.deleteOne(myquery, function(err, obj) {
    if (err) throw err;
    console.log(result.name);
   });
```

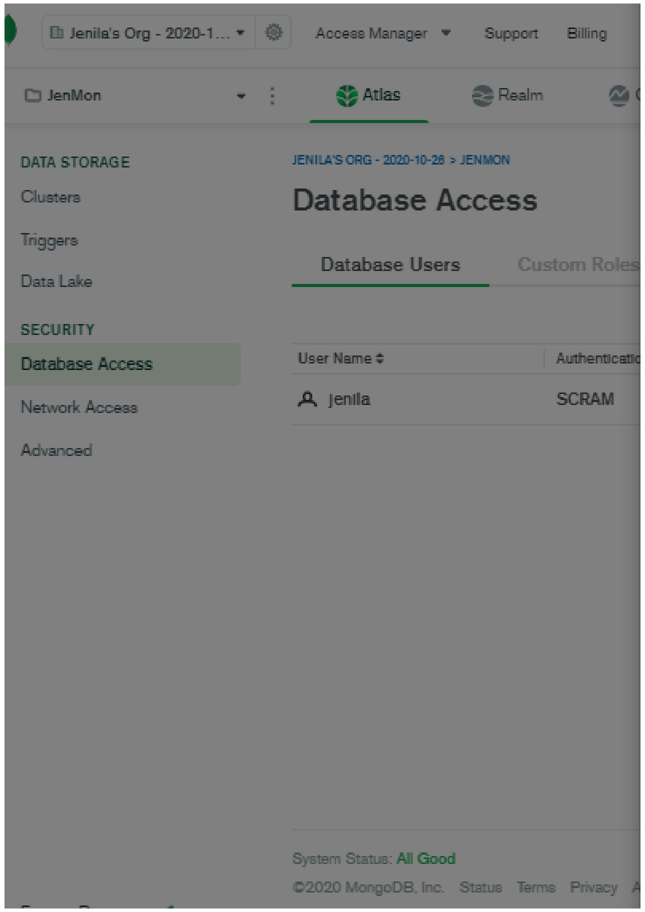
Delete Multiple Records

Delete all documents were the address starts with the letter “C":

```
//Delete multiple records:
   var myquery = { address: /^C/ };
  collection.deleteMany(myquery, function(err, obj) {
   if (err) throw err;
   console.log(result.name);
  });
```

Part 2: MongoDB database (cloud)

Download Free MongoDB database from https://www.mongodb.com.



JENILA'S ORG - 2020-10-26 > JENMON

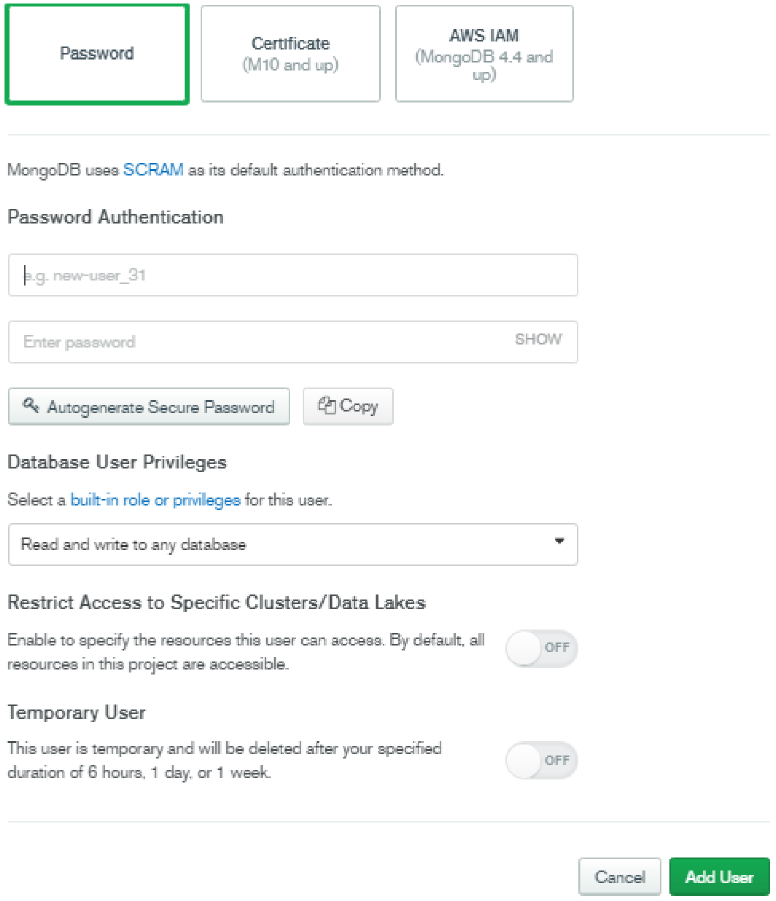
Database Access

Database Users | Custom Roles

User Name	Authenticatio
jenila	SCRAM

System Status: All Good

©2020 MongoDB, Inc. Status Terms Privacy



MongoDB uses SCRAM as its default authentication method.

Password Authentication

e.g. new-user_31

Enter password SHOW

Database User Privileges

Select a built-in role or privileges for this user.

Read and write to any database

Restrict Access to Specific Clusters/Data Lakes

Enable to specify the resources this user can access. By default, all resources in this project are accessible. OFF

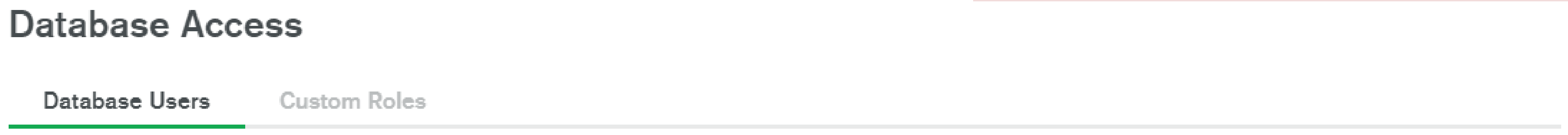
Temporary User

This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week. OFF

Step 1: Add User

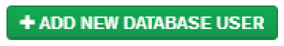
Step 1.1: Select database Access Menu
Step 1.2: Click Add New user database User

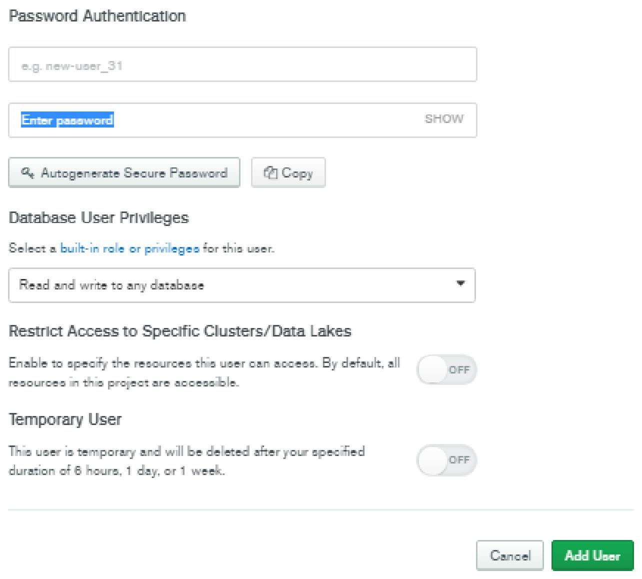
Database Access->Add New Database User



Database Access

Step 1.3: Enter username, Password and click Add User button





Password Authentication

e.g. new-user_31

Enter password SHOW

Database User Privileges

Select a built-in role or privileges for this user.

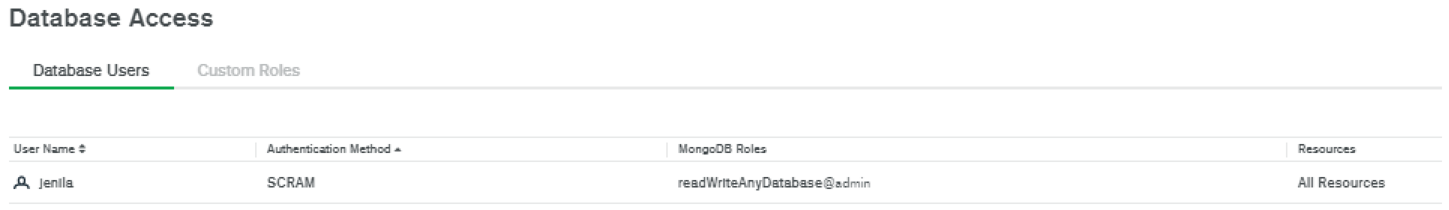
Read and write to any database

Restrict Access to Specific Clusters/Data Lakes

Enable to specify the resources this user can access. By default, all resources in this project are accessible. OFF

Temporary User

This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week. OFF



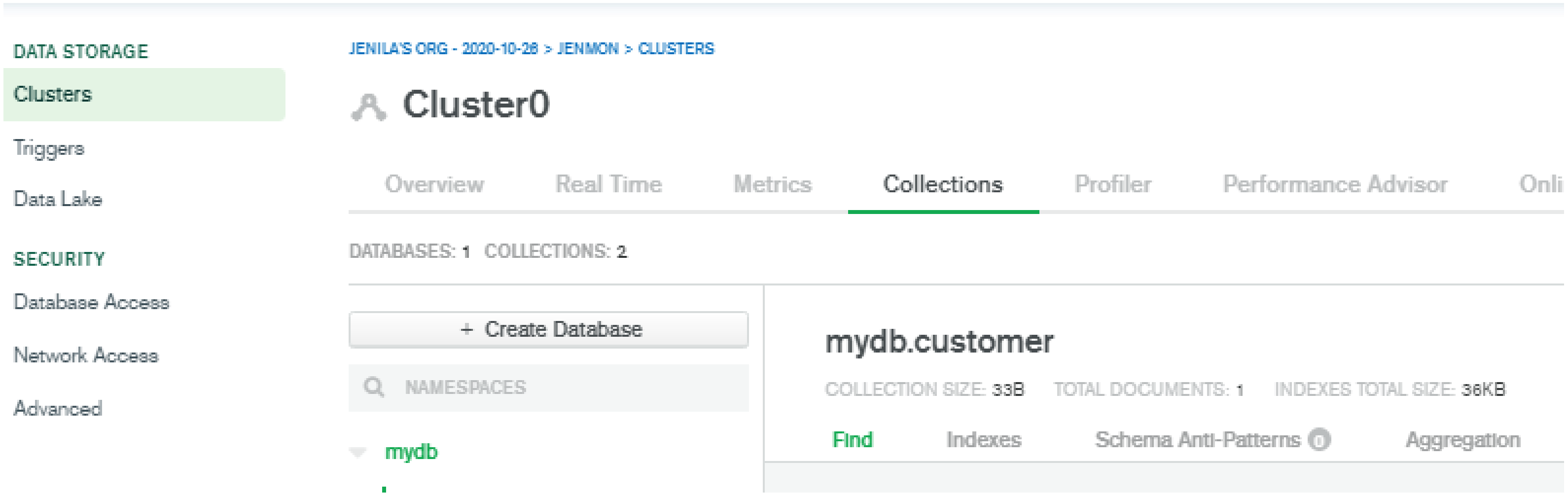
Database Access

User Name	Authentication Method	MongoDB Roles	Resources
jenila	SCRAM	readWriteAnyDatabase@admin	All Resources

Step 2:Create Database & Collection

Step 2.1: Select Clusters menu -> Collections tab/ Collections button

Step 2.2: Click Create Database

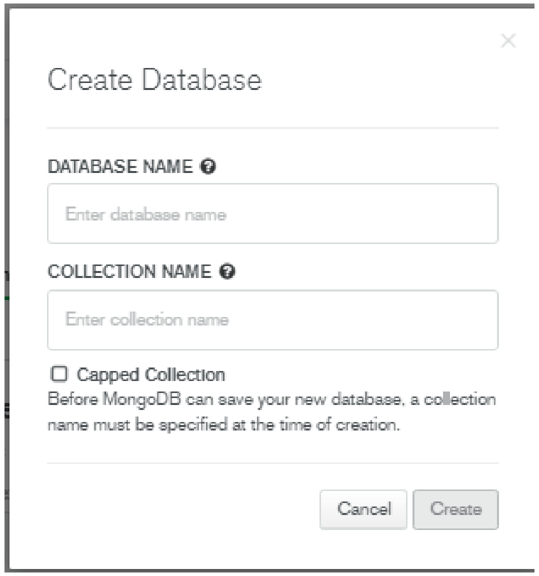


DATABASES: 1 COLLECTIONS: 2

mydb.customer

COLLECTION SIZE: 33B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 36KB

Step 2.3: Type Database Name and Collection Name

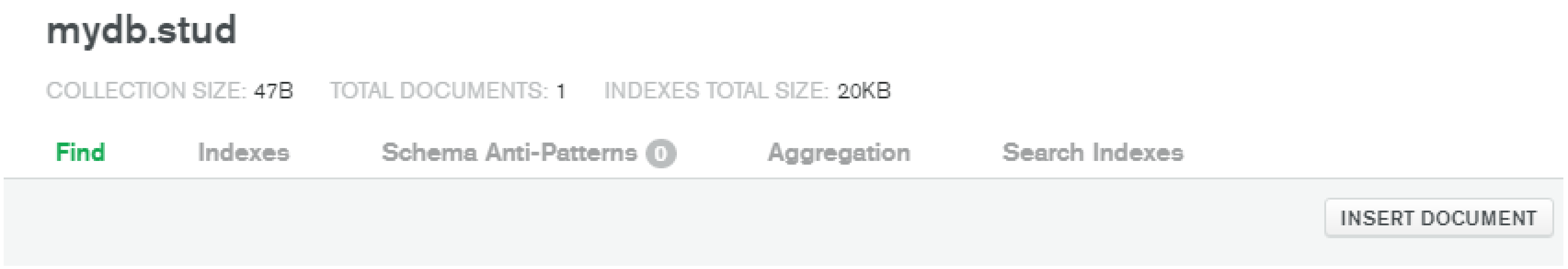


☐ Capped Collection

Before MongoDB can save your new database, a collection name must be specified at the time of creation.

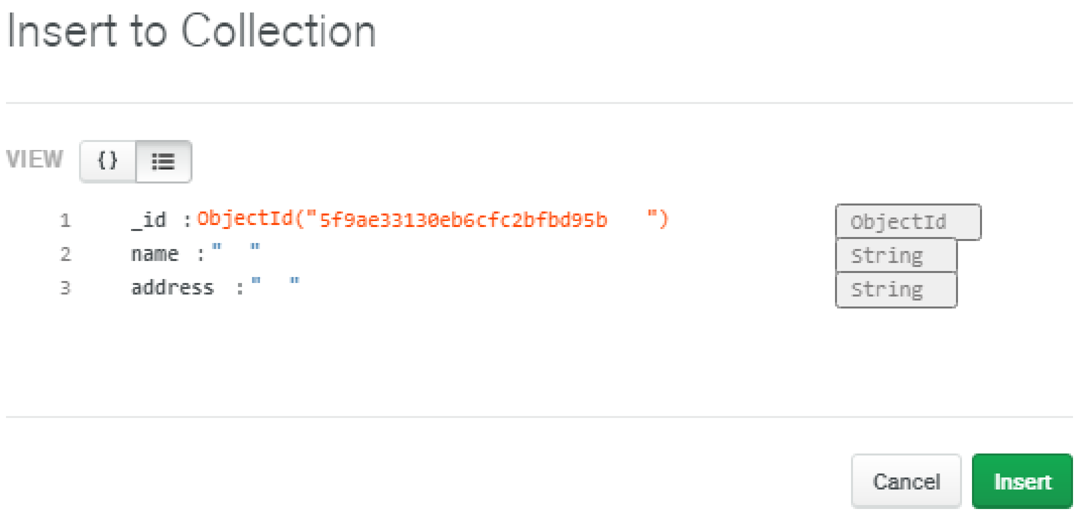
Step 3:Create Document

Step 3.1: Select your Database and collection
Step 3.2: Click Insert Document


mydb.stud

COLLECTION SIZE: 47B   TOTAL DOCUMENTS: 1   INDEXES TOTAL SIZE: 20KB

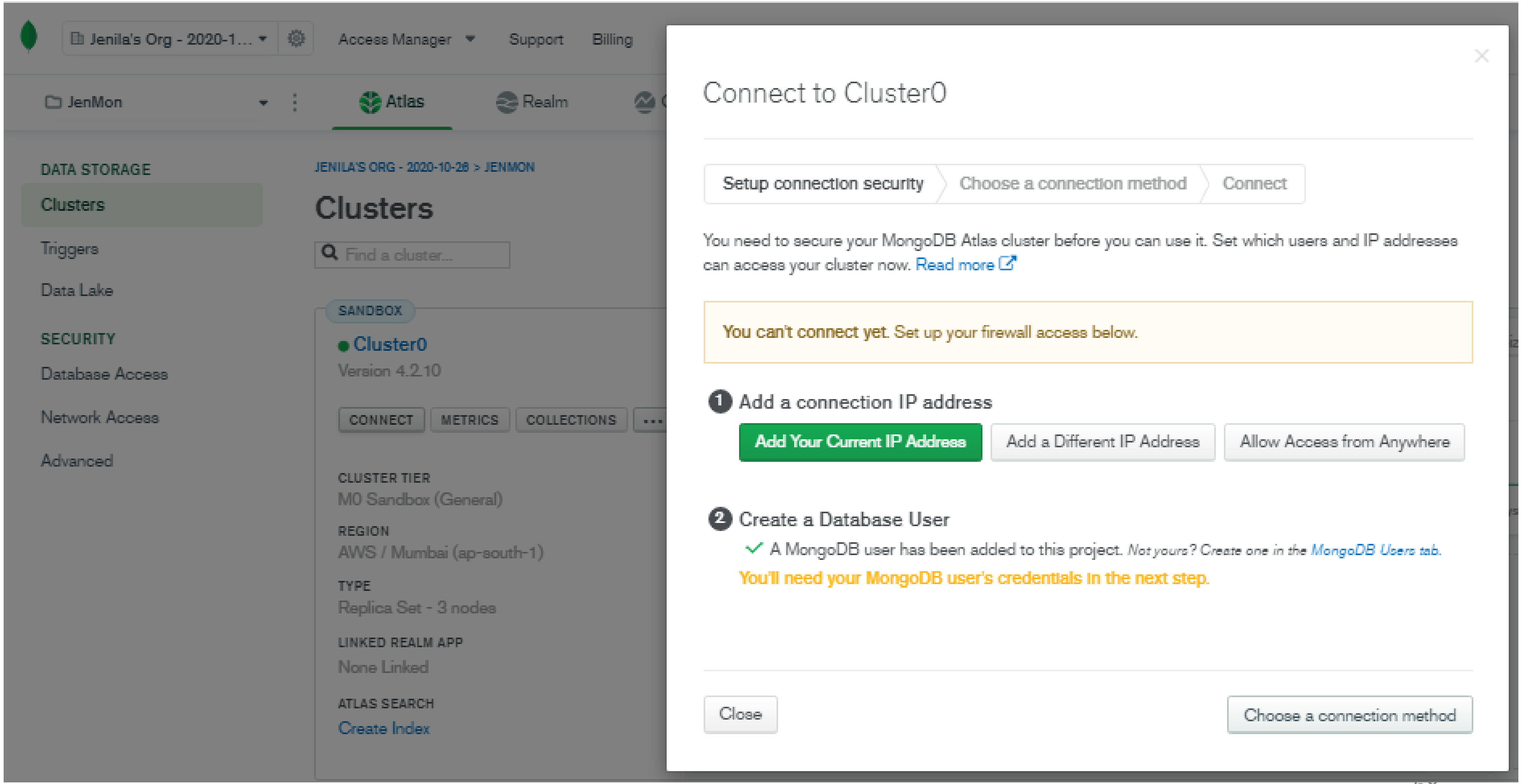
Step 3.3: Insert Attributes


Insert to Collection

```
_id  :ObjectId("5f9ae33130eb6cfc2bfbd95b   ")     ObjectId
name  :"   "                                     String
address  :"   "                                  String
```

Step 4: Connect to Cluster

Step 4.1: Select Clusters menu -> Click Connect button
Step 4.2: Click Choose a connection method



Connect to Cluster0

Setup connection security | Choose a connection method | Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. Read more

You can't connect yet. Set up your firewall access below.

1 Add a connection IP address

Add Your Current IP Address | Add a Different IP Address | Allow Access from Anywhere

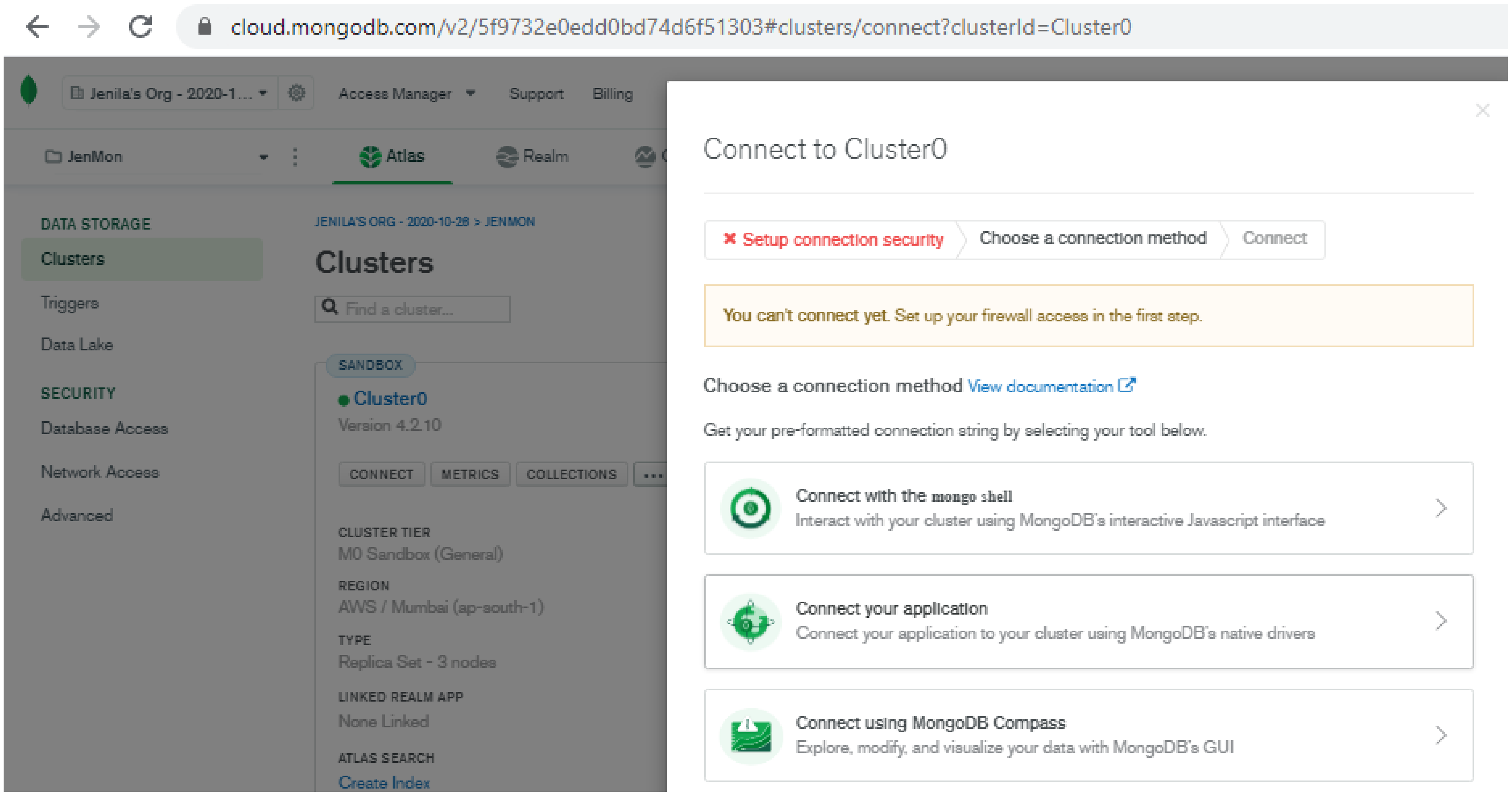
2 Create a Database User

✓ A MongoDB user has been added to this project. *Not yours? Create one in the MongoDB Users tab.*

You'll need your MongoDB user's credentials in the next step.

Close | Choose a connection method

Step 4.3: Click Connect your application


cloud.mongodb.com/v2/5f9732e0edd0bd74d6f51303#clusters/connect?clusterId=Cluster0

Connect to Cluster0

✖ Setup connection security | Choose a connection method | Connect

You can't connect yet. Set up your firewall access in the first step.

Choose a connection method View documentation

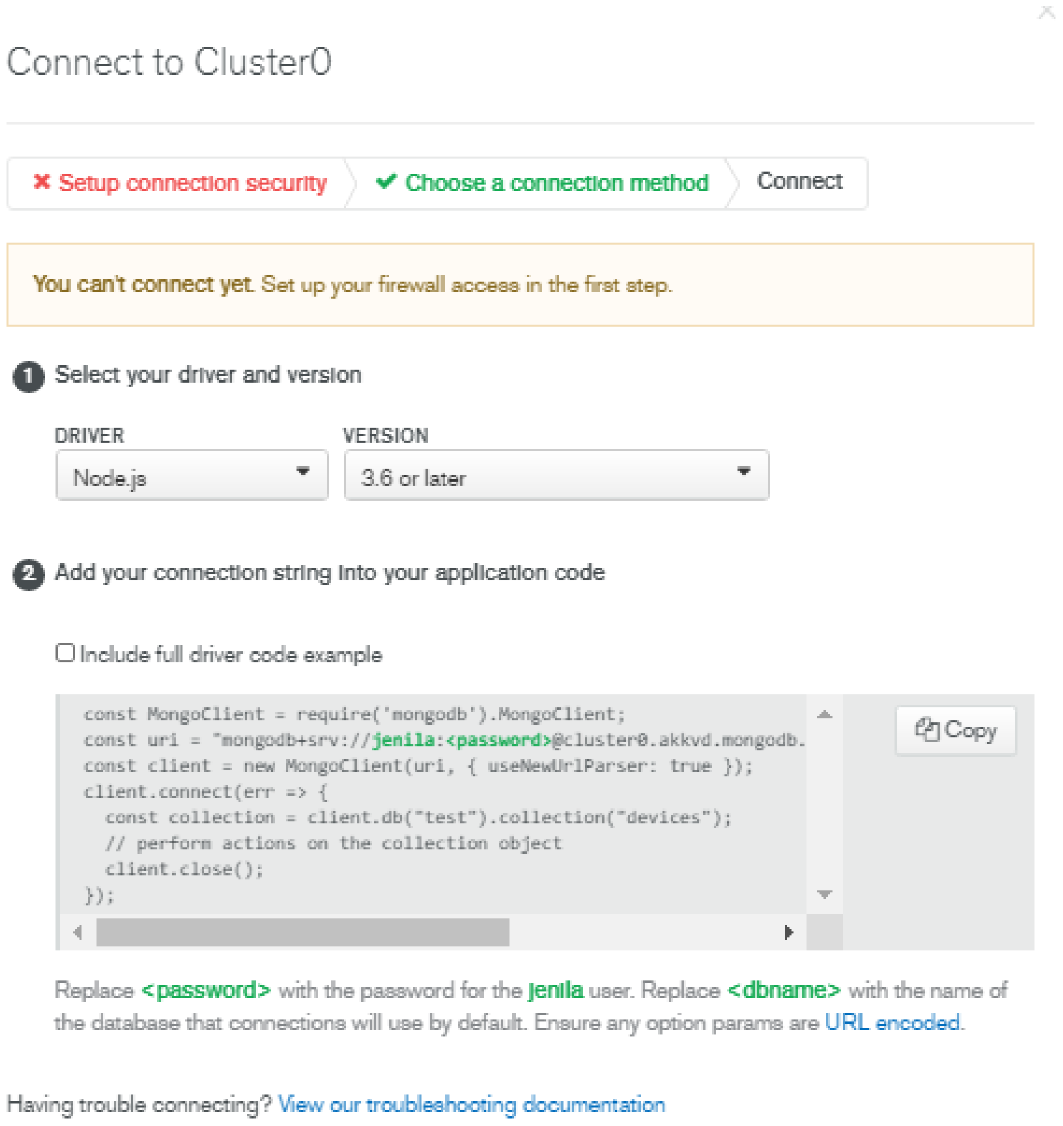
Get your pre-formatted connection string by selecting your tool below.

Connect with the mongo shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

Step 4.4: Select Node.js
Step 4.5: Check Include full driver code example
Step 4.6: Copy the code



Connect to Cluster0

You can't connect yet. Set up your firewall access in the first step.

1 Select your driver and version

DRIVER: Node.js
VERSION: 3.6 or later

2 Add your connection string into your application code

☐ Include full driver code example

```
const MongoClient = require('mongodb').MongoClient;
const uri = "mongodb+srv://jenila:<password>@cluster0.akkvd.mongodb.
const client = new MongoClient(uri, { useNewUrlParser: true });
client.connect(err => {
  const collection = client.db("test").collection("devices");
  // perform actions on the collection object
  client.close();
});
```

Replace <password> with the password for the jenila user. Replace <dbname> with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Having trouble connecting? View our troubleshooting documentation

MongoDB Connection code

- //Change the code

```
const MongoClient = require('mongodb').MongoClient;
const uri =
"mongodb+srv://jenila:<password>@cluster0.akkvd.mongodb.net/<dbname>?retryWrites=true&w=majority";
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });
client.connect(err => {
  const collection = client.db("test").collection("devices");
  // perform actions on the collection object
 // type your code here

  client.close();
});
```

Example: Insert a Single record

```
const MongoClient = require('mongodb').MongoClient;
const uri =
"mongodb+srv://jenila:jenila@cluster0.akkvd.mongodb.net/mydb?retryWrite
s=true&w=majority";
const client = new MongoClient(uri,
{ useNewUrlParser: true,
useUnifiedTopology: true });
client.connect(err => {
 const collection = client.db("mydb").collection("student");
 // perform actions on the collection object
var myobj = [
  { name: 'John', address: Chennai 71'},
];
 collection.insertOne(myobj);
 console.log('Object Inserted')
 client.close();
});
```

References

- Brad Dayley, Brendan Dayley, and Caleb Dayley, Node.js, MongoDB and Angular Web Development: The definitive guide to using the MEAN stack to build web applications, 2 nd Edition, Pearson Education, 2018
- https://www.w3schools.com/nodejs/nodejs_mongodb_insert.asp