CSCI3100 Project Tutorial 4
Server-side Technologies

Yun PENG

csci3100@cse.cuhk.edu.hk

Department of Computer Science and Engineering

The Chinese University of Hong Kong

Outline

- Introduction to Web Development
  - Socket
  - Protocols (TCP/UDP/HTTP)
  - Express Framework
  - Database in Node.js
  - Node.js IDE: WebStorm
  - Django, Node.js & Flask

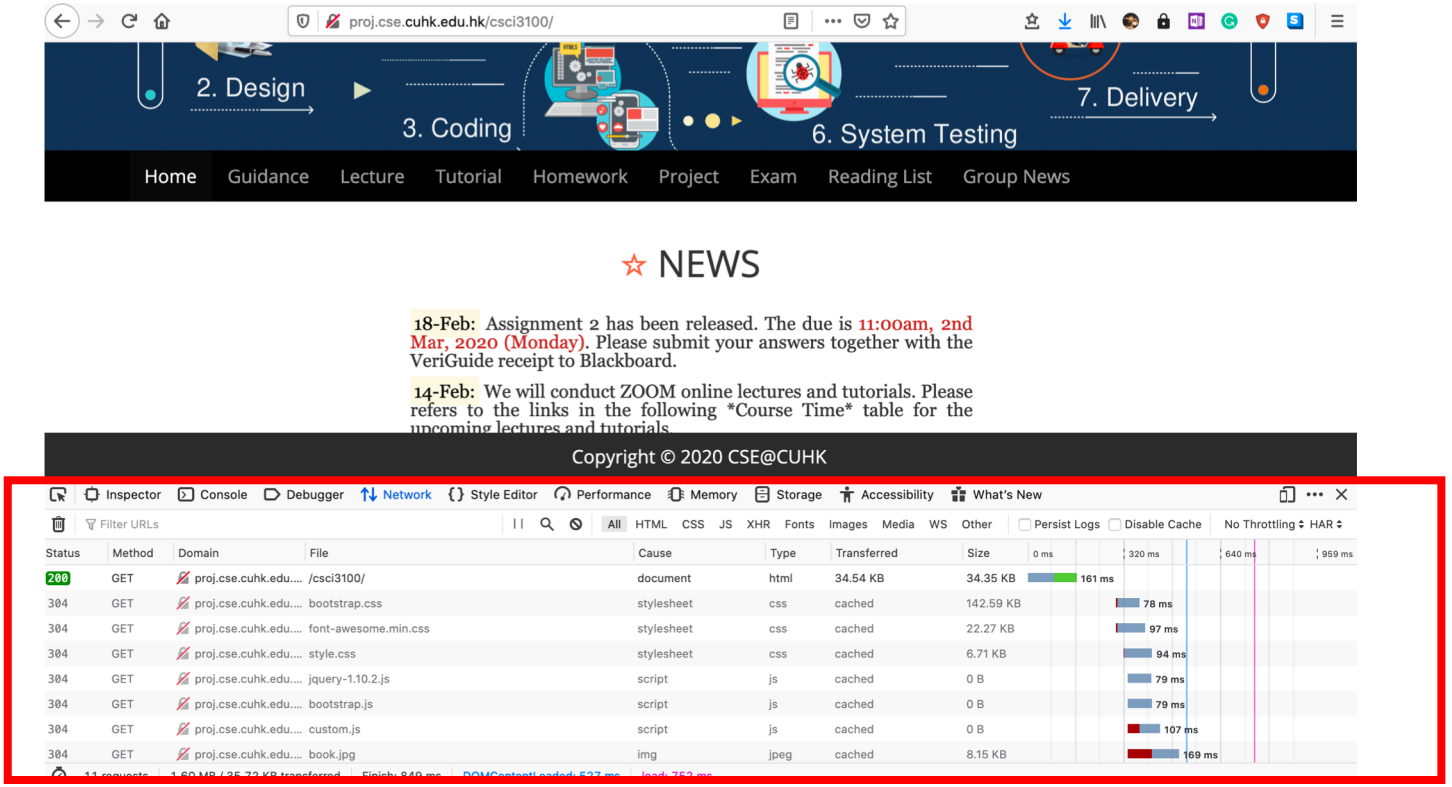
Outline

- Introduction to AWS
  - Amazon and AWS
  - Create Micro Instance on EC2
  - Basic shell commands
- Database Basics
  - Relational Database (MySQL)
  - NoSQL Database (MongoDB)
- Reference

Access to a website

Chrome Menu->View->Developer->Developer Tools

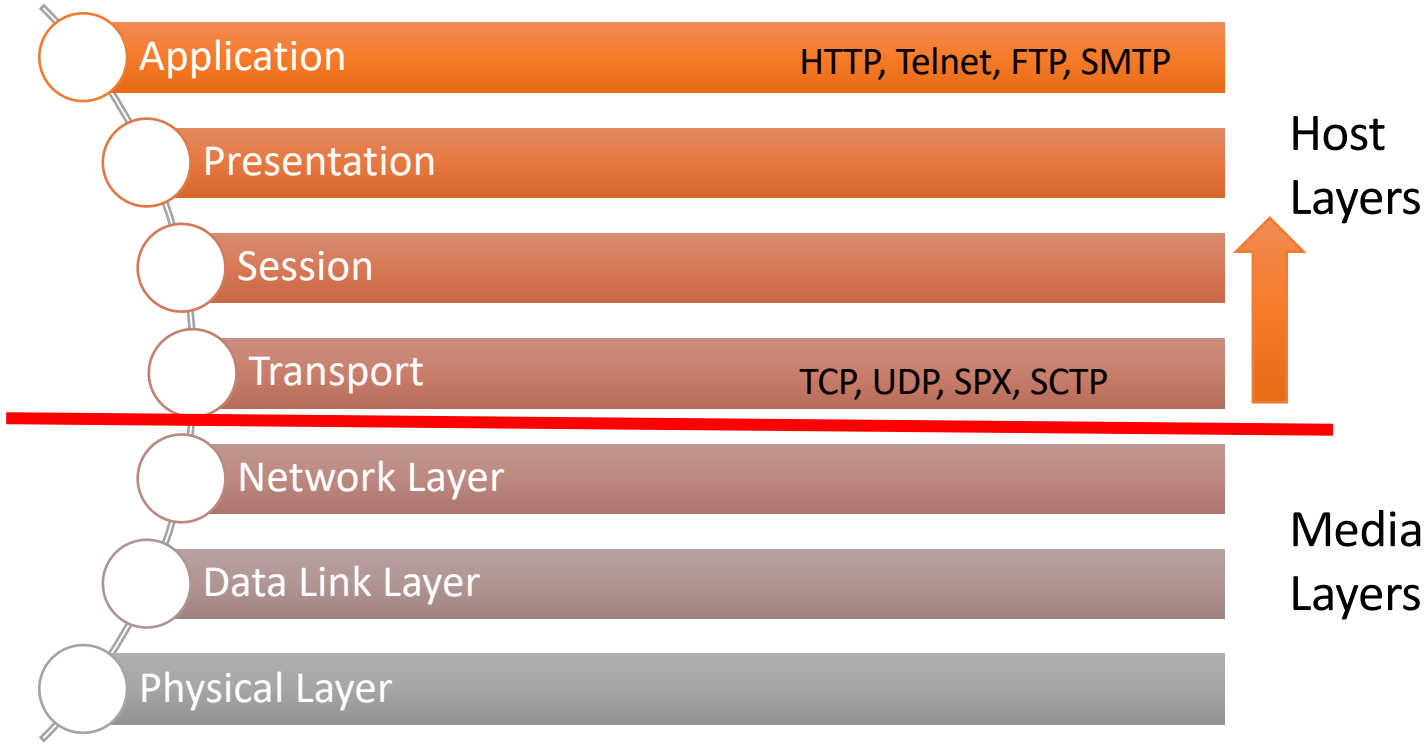
Firefox Menu->Web Developer->Toggle Tools



How browsers work

- What happened when you visit a webpage?
  - The browser *communicates* with a **process** in the remote server
  - The remote process *send* data (HTML, etc.) back
  - The browser *renders* (display) the data
- In detail
  - How Browsers Work: Behind the scenes of modern web browsers

OSI (Open Systems Interconnection) Model



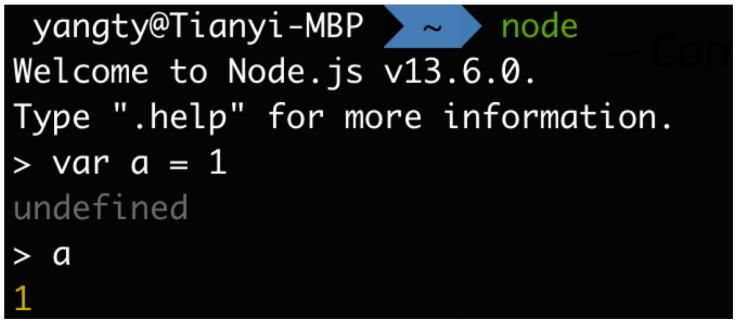
Motivation of Using Node.js

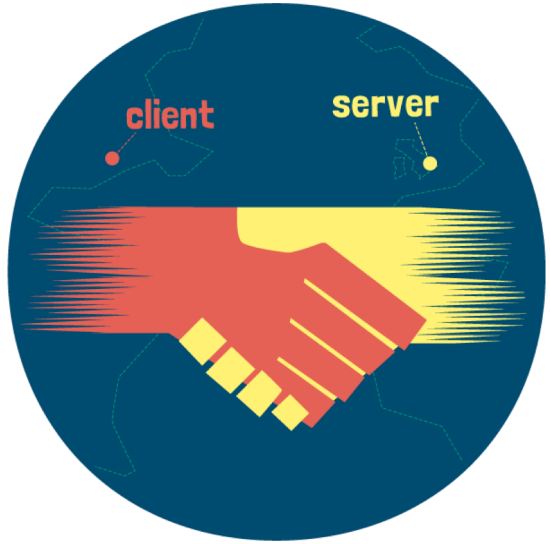
- Save time to learn new languages
  - Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine
- Node.js is very efficient
  - event-driven, non-blocking I/O model
- Learn new paradigm to catch up the tendency of Web development
- Using other frameworks are also accepted
  - Will not affect your grade

Installation

- Follow: https://nodejs.org
- After installation:
  - Test the version:

  - Command line interface:




client
server


TCP/IP
Transmission Control Protocol/Internet Protocol

Transmission Control Protocol (TCP)

```
var net = require("net");          // Get "Net" Module

var server = net.createServer({
    allowHalfOpen: false
}, function(socket) {
// handle connection
    socket.end("Hello and Goodbye!\n");
});

server.listen(8000, "127.0.0.1");      // Register the address and port
```

Demo of TCP Connection

$ telnet localhost 8000

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

Hello and Goodbye!

Connection closed by foreign host.

p.s: enabling ‘telnet’ in win10 with cmd (administrator):
$ dism /online /Enable-Feature /FeatureName:TelnetClient
https://www.rootusers.com/how-to-enable-the-telnet-client-in-windows-10/

For mac, it should support telnet by default, but if your macOs is high Sierra(10.13), you should refer to: https://medium.com/ayuth/bring-telnet-back-on-macos-high-sierra-11de98de1544

User Datagram Protocol (UDP)-Server

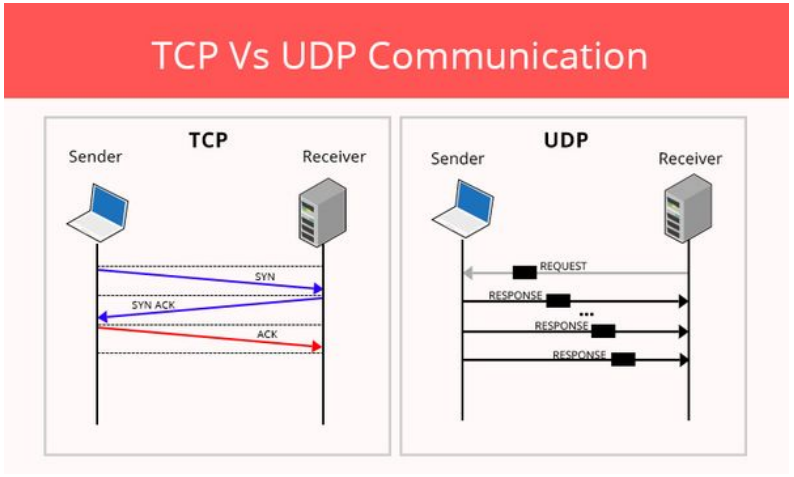
```
var dgram = require("dgram");      // Get"dgram"module
var server= dgram.createSocket("udp4", function(msg,
rinfo) {
   console.log("received " + rinfo.size + " bytes");
   console.log("from " + rinfo.address + ":" + rinfo.port);
   console.log("message is: " + msg.toString());
});

server.bind({address: "localhost", port: 8012}, function() {
   console.log("bound to ");
   console.log(server.address());
});
```

Sending Data via UDP-Client

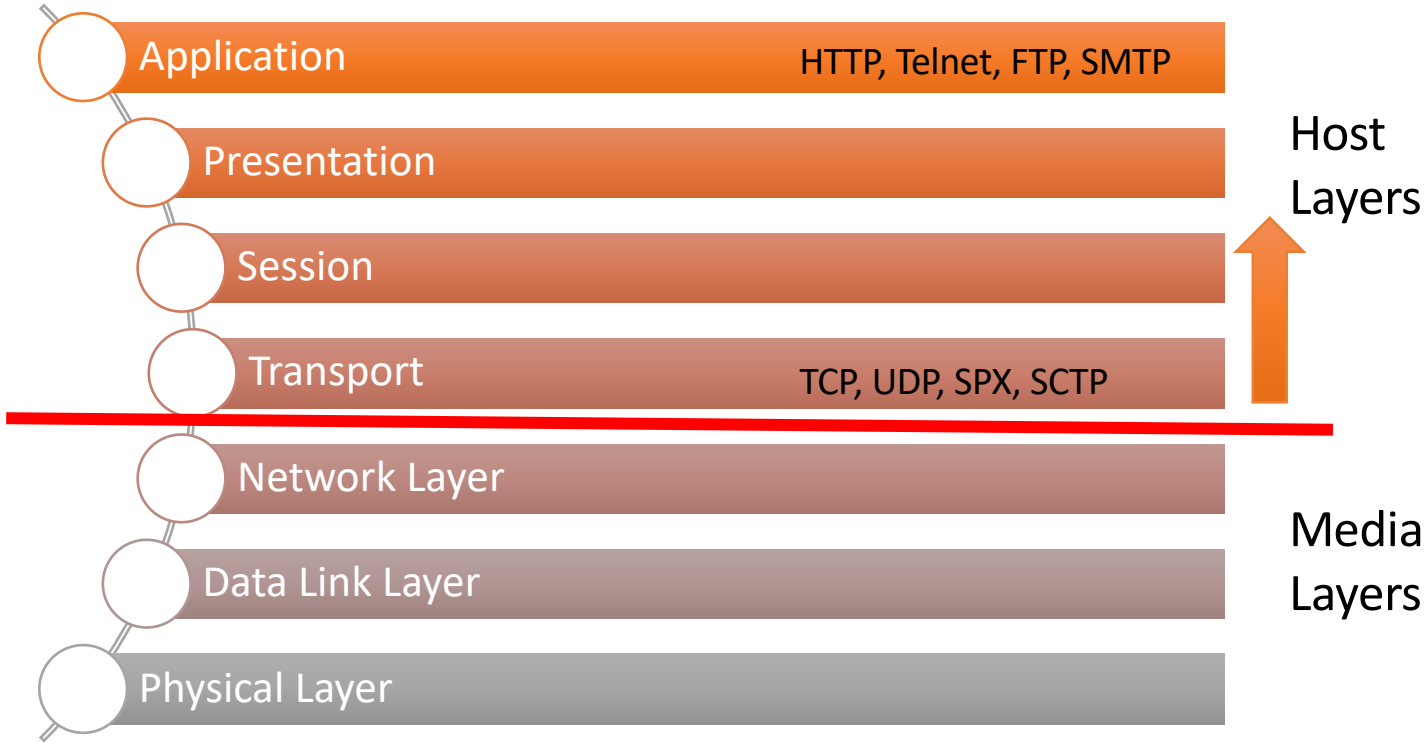
```
var dgram = require("dgram");
var client = dgram.createSocket("udp4");
var message = new Buffer("Hello UDP");

client.send(message, 0, message.length, 8012,
"127.0.0.1", function(error, bytes) {
    if (error) {
        console.error("An error occurred while sending");
    } else {
        console.log("Successfully sent " + bytes + "
bytes");
    }
    client.close();
});
```

TCP vs UDP

TCP	UDP
Reliable	Unreliable
Numbers each data packet	Does not keep data packet in order
Confirms receipt of the data packet	Does not confirm data packet receipt
Include error-check capabilities to prevent data corruption	Maximize performance
Consumes more network bandwidth	Consumes less bandwidth
Overall slower processing	Faster processing
three-way handshake	no



OSI (Open Systems Interconnection) Model



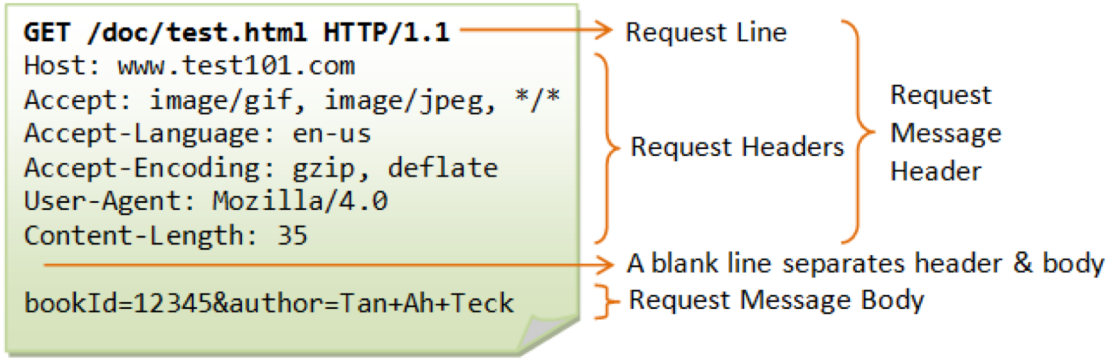
HyperText Transfer Protocol (HTTP)

- Higher layer than TCP and UDP

```
var http = require("http");
var server = http.createServer(function(request,
response) {
    response.write("Hello HTTP!");
    response.end();
});
server.listen(8000);
```

Open 127.0.0.1:8000 in browser

HTTP Request & Request Method



GET	**A GET request should not alter the state of the server and is essentially a read operation.**
POST	POST requests are the submission of HTML forms and the addition of data to a database.
HEAD	Retrieves the same data as an equivalent GET request, except the response body should be omitted.
PUT	PUTs are used to update existing resources on the server.
DELETE	Used to delete a resource from a server.
CONNECT	Used to create a tunnel through a proxy server.

Get the Info of Incoming Connection

```
var http = require("http");
var server = http.createServer(function(request,
response) {
   var requestLine = request.method + " " + request.url +
      " HTTP/" + request.httpVersion;
   console.log(requestLine);
   response.end();
});
server.listen(8000);
```

Request Header

Accept	Specifies the Content-Types that the client is willing to accept for this request.
Accept-Encoding	Provides a list of acceptable encodings
Cookie	Small pieces of data that the server stores on the client
Content-Length	The length of the request body in octets.
Host	The domain and port of the server
User-Agent	A string identifying the type of client

```
var http = require("http");
http.createServer(function(request, response) {
    console.log(request.headers);
    response.end();
}).listen(8000);
```

Response Code

200	OK	Indicates that the HTTP request was handled successfully
201	Created	Indicates that the request has been fulfilled, and a new resource has been created on the server
301	Moved Permanently	The requested resource has permanently moved to a new URL
400	Bad Request	Indicates that the request was malformed and could not be understood.
404	Not Found	The server could not locate the requested URL
500	Internal Server Error	The server encountered an error while attempting to fulfill the request

Demo of Response Code

```
var http = require("http");
http.createServer(function(request, response) {
    if (request.url === "/foo") {
        response.end("Hello HTTP");
    } else {
        response.statusCode = 404;
        response.end();
    }
}).listen(8000);
```

Response Header

Cache-Control	Specifies whether a resource can be cached
Content-Type	Specifies the MIME type of the response body
WWW-Authenticate	If an authentication scheme is implemented for a given resource, this header is used to identify the scheme
Content-Encoding	Specifies the encoding used on the data

```
var http = require("http");
var server = http.createServer(function(request, response) {
    response.setHeader("Content-Type", "text/html");
    response.write("Hello <strong>HTTP</strong>!");
    response.end();
});
server.listen(8000);
```

Multiple Resources

```
var http = require("http");
http.createServer(function(request, response) {
   if (request.url === "/" && request.method === "GET") {
      response.writeHead(200, {
         "Content-Type": "text/html"
      });
      response.end("Hello <strong>home page</strong>");
   } else if (request.url === "/bar" && request.method === "GET") {
      response.writeHead(200, {
         "Content-Type": "text/html"
      });
      response.end("Hello <strong>bar</strong>");
   } else {
      response.writeHead(404, {
         "Content-Type": "text/html"
      });
      response.end("404 Not Found");
   }
}).listen(8000);
```

Terrible?


node
JS
™
express

A Simpler Solution

```
var express = require("express");
var http = require("http");
var app = express();
app.get("/", function(req, res, next) {
    res.send("Hello <strong>home page</strong>");
});
app.get("/bar", function(req, res, next) {
    res.send("Hello <strong>bar</strong>");
});
http.createServer(app).listen(8000);
```

Note: Install Express Framework using “npm install express”

Route Parameter

```
var express = require("express");
var http = require("http");
var app = express();
app.get(/\/products\/([^\/]+)\/?$/, function(req, res,
next) {
   res.send("Requested " + req.params[0]);
});
http.createServer(app).listen(8000);
```

PS: must use a JS regular expression object instead if string by wrapping your regular expression in slashes (/) instead of quotes (")
https://www.kevinleary.net/regex-route-express/

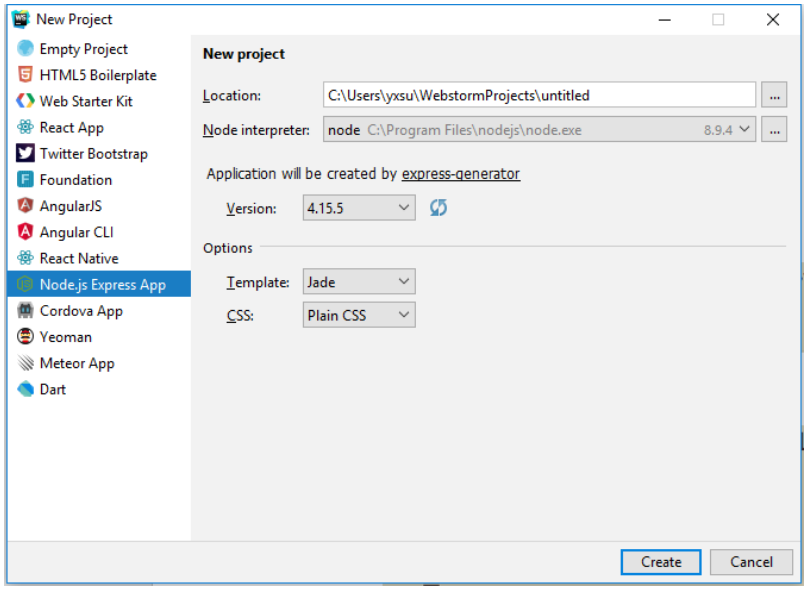
Route with Named Parameter

```
var express = require("express");
var http = require("http");
var app = express();
app.get("/products/:productId", function(req, res, next)
{
   res.send("Requested " + req.params.productId);
});
http.createServer(app).listen(8000);
```

Creating an Express Framework

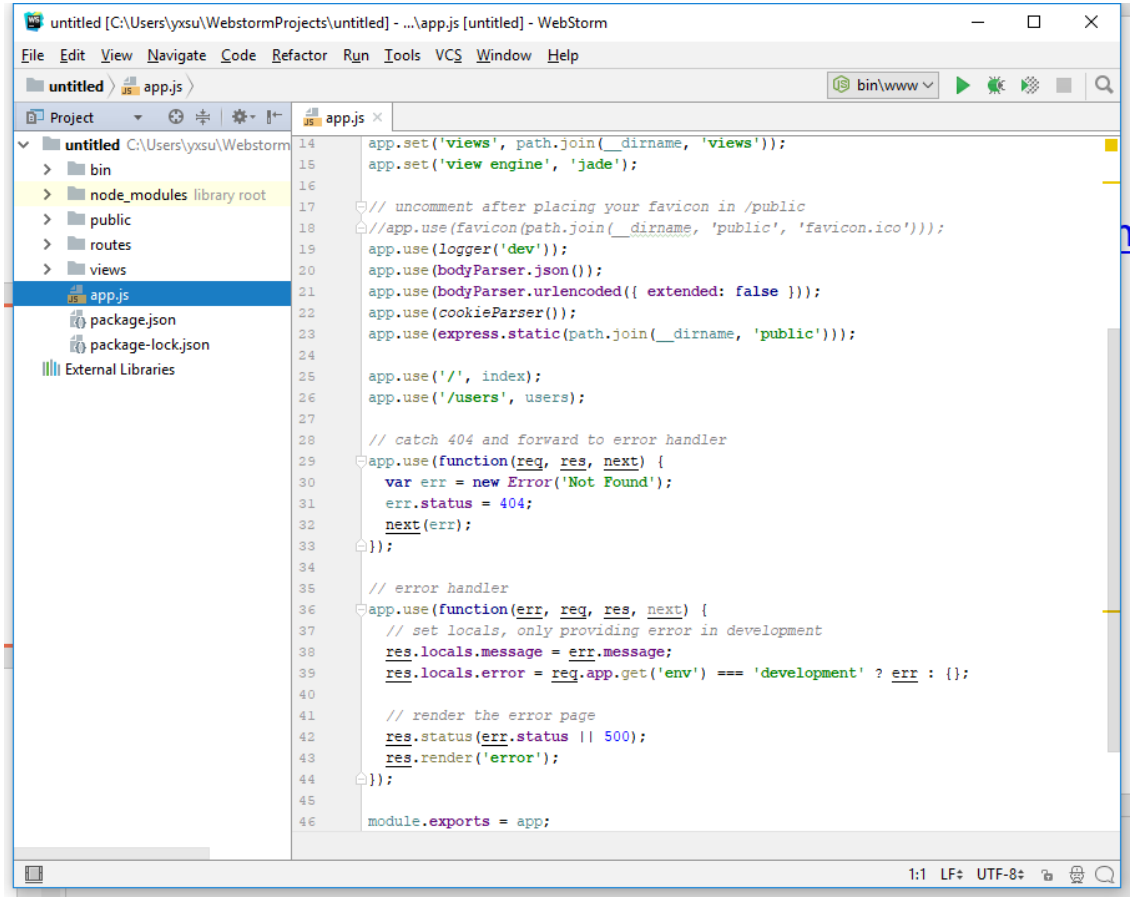
- Easiest way: WebStorm (demo)
- Or:
  - Installation:
    - $npm install -g express
  - Create:
    - $ express testapp
  - Install dependency:
    - cd testapp && npm install

WebStorm (IDE)



- https://www.jetbrains.com/webstorm/
- Apply for student account:
  - https://www.jetbrains.com/student/

WebStorm (IDE)



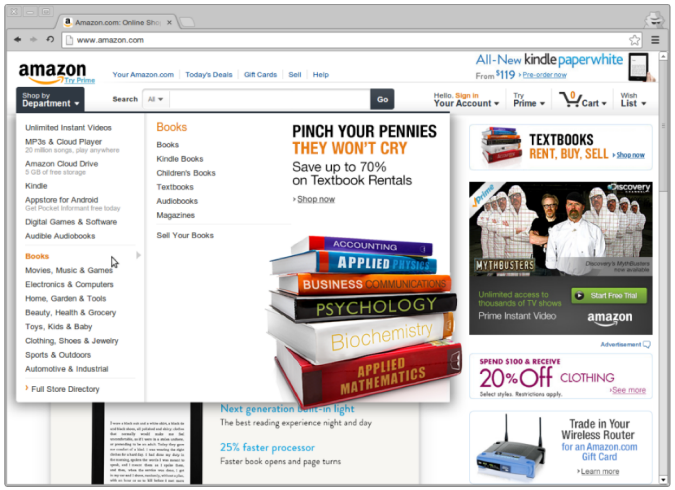
NodeJS, Django & Flask

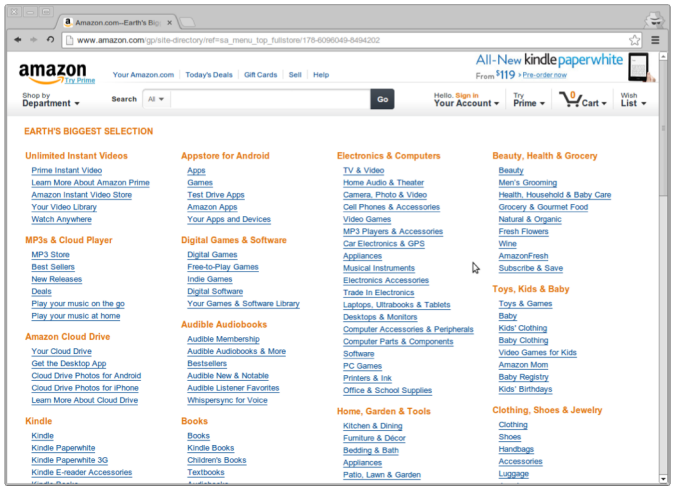
JavaScript | Python | Python

	Node.js	Django
Pros	• Availability of great libraries. • High performance. • Huge user community. • Handles concurrent requests easily.	• Works fine with relational databases. • Easy to learn and very scalable. • Speedy development process. • Huge user community and great documentation.
Cons	○ Asynchronous programming is difficult ○ Not great with CPU intensive apps (single thread) ○ Callbacks cause many nested callbacks.	○ Not great for small-scale apps. ○ A full understanding of the framework is needed.

Amazon

- Amazon is one of the major e-commerce player in the US
  - It starts as an online bookstore
  - It sells various kind of things nowadays





Amazon

- How a traditional e-commerce company end up selling their computing power?
  - Traffic to Amazon erupts significantly on Christmas/Black Friday (Think about Taobao on Nov. 11)
  - Amazon must build their computational infrastructure powerful enough to cope with the peak traffic
  - Most of the computing power are wasted during other time

AWS

- AWS stands for Amazon Web Services
  - AWS is one of the major *cloud computing* environment
  - Bignames in startup are powered by AWS
    - Reddit, Pinterest, Flipboard, Foursqure
  - Bigger names use their own cloud
    - Microsoft, Facebook, Google, Alibaba, Tencent
  - What exactly is AWS
    - http://www.youtube.com/watch?v=jOhbTAU4OPI

AWS EC2

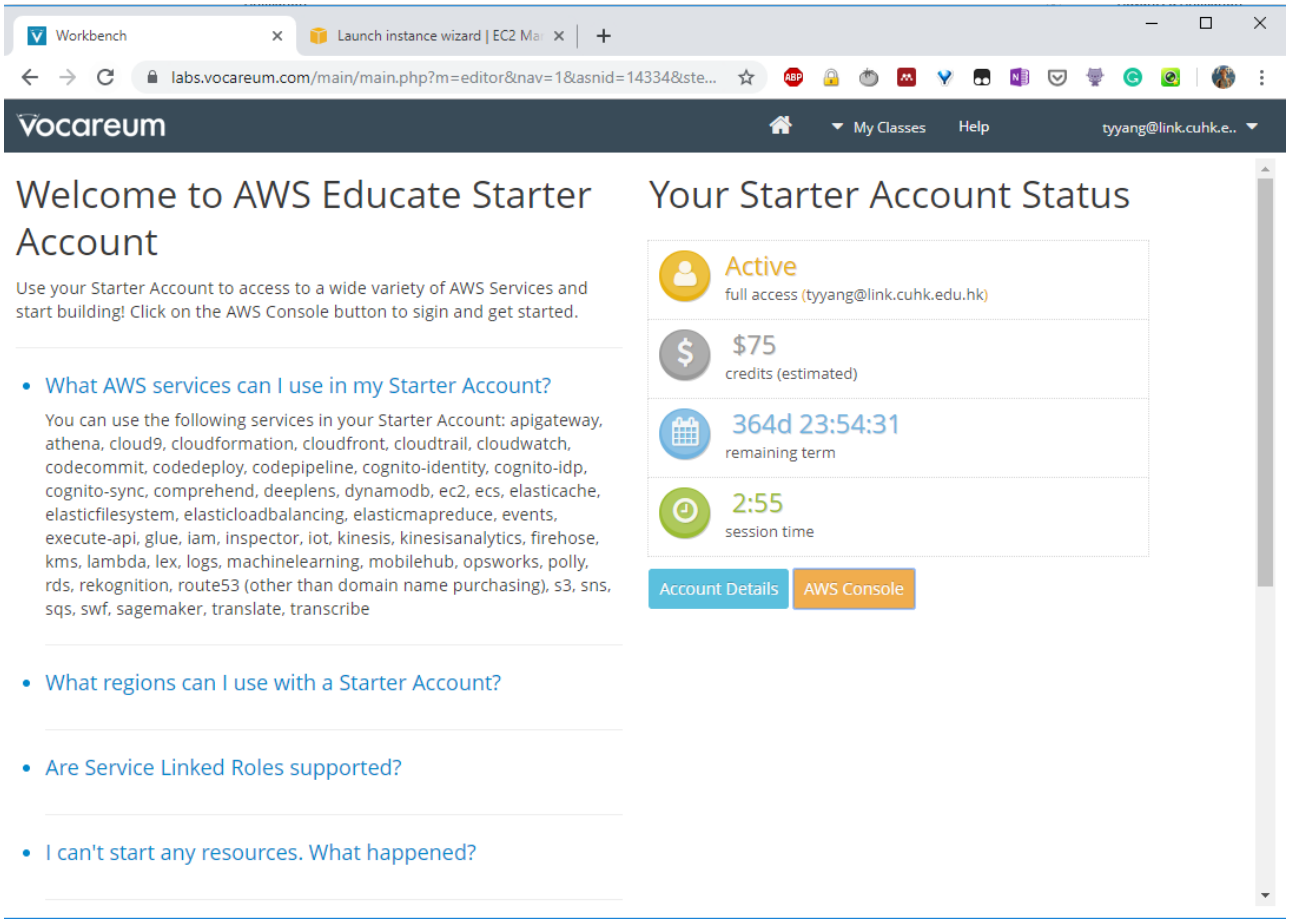
- Amazon EC2 stands for Elastic Computing Cloud
  - For this tutorial, think of it as a single virtual machine
- You are encouraged to deploy your project backend on EC2 to allow public access

Create AWS Account

- Create an account on AWS (If you don't have)
  - AWS Educate: no credit card needed
    - Students at AWS Educate member institutions will receive up to US$75 of AWS credit per year in their AWS Educate Starter Account
    - https://www.awseducate.com/registration#APP_TYPE
  - AWS Personal: credit card needed.
    - New user are eligible for free access to micro instance for 12 months, 750 hours per month

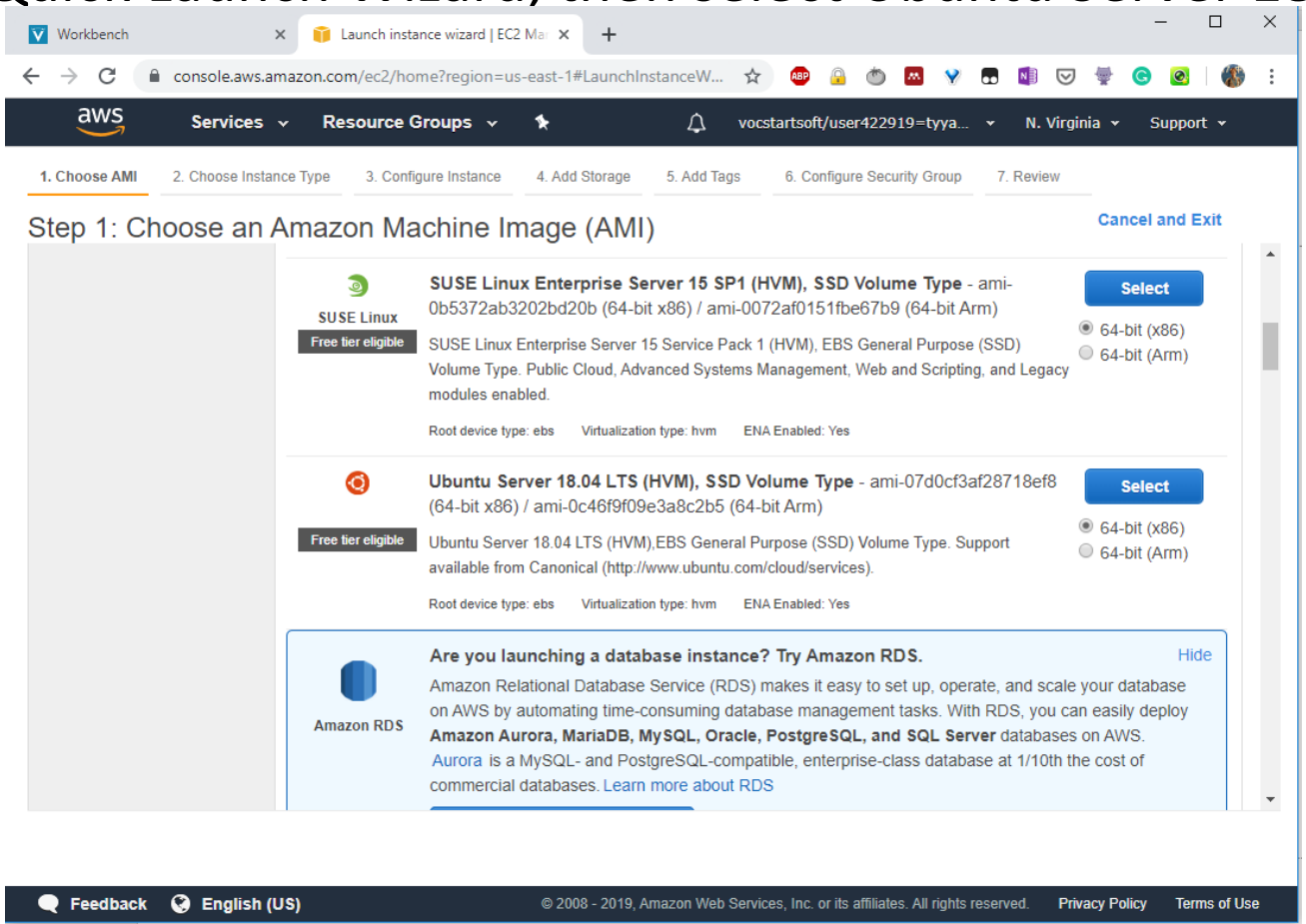
Create Micro Instance on EC2

1. Login to your account and go to the AWS console



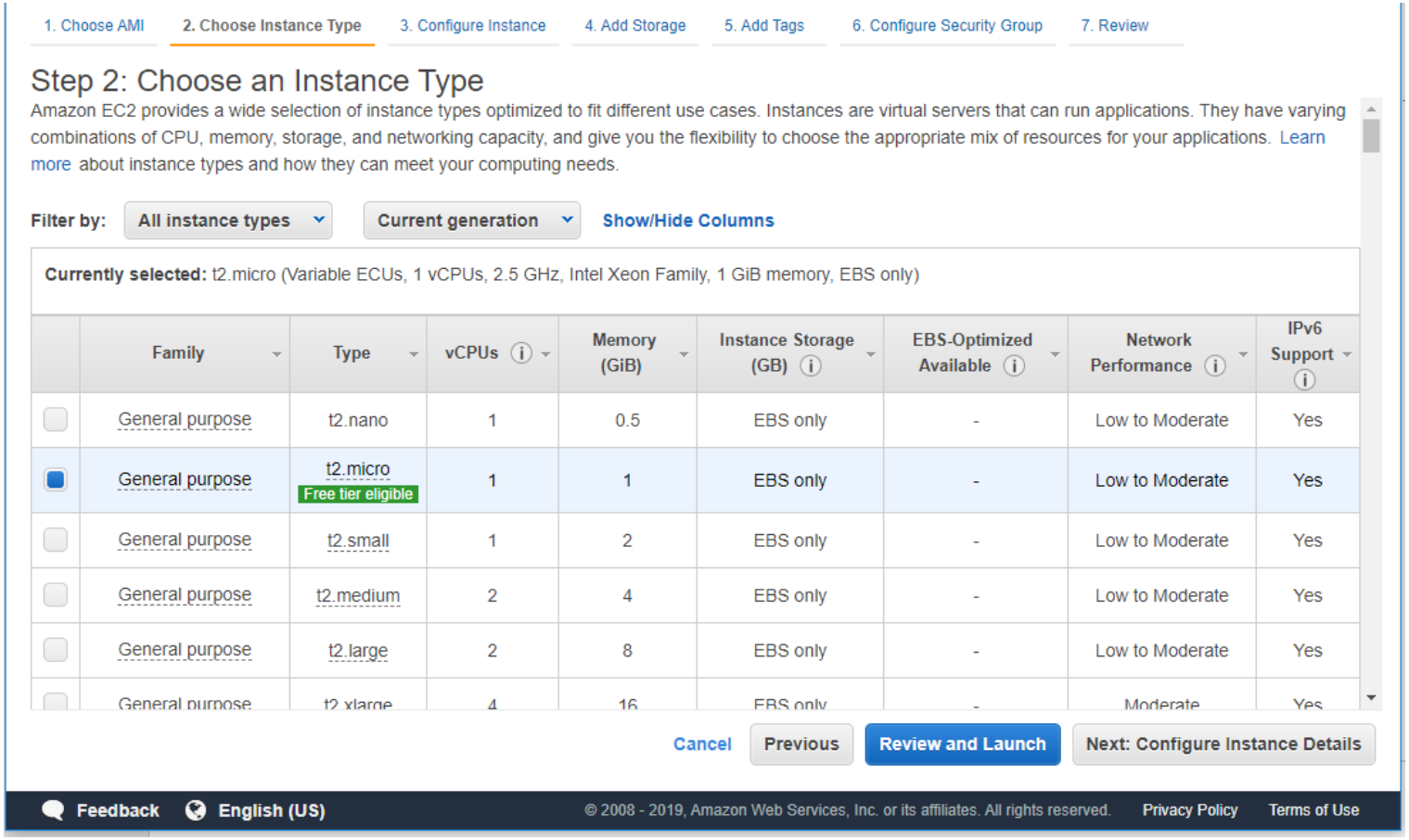
Create Micro Instance on EC2

2. Click the “Launch a virtual machine” to create an EC2 Instance
Select Quick Launch Wizard, then select Ubuntu Server 18.04



Create Micro Instance on EC2

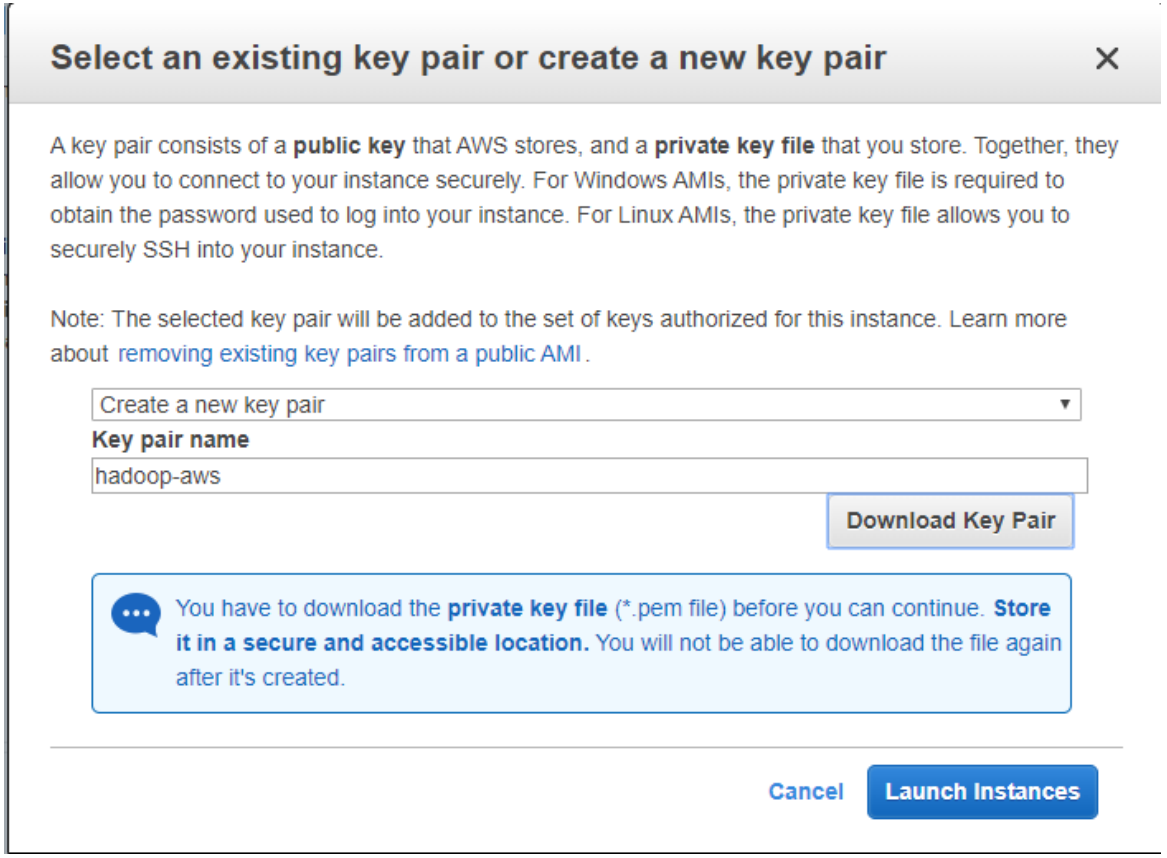
For step 2 – 7 : use default option, then click launch



Create Micro Instance on EC2

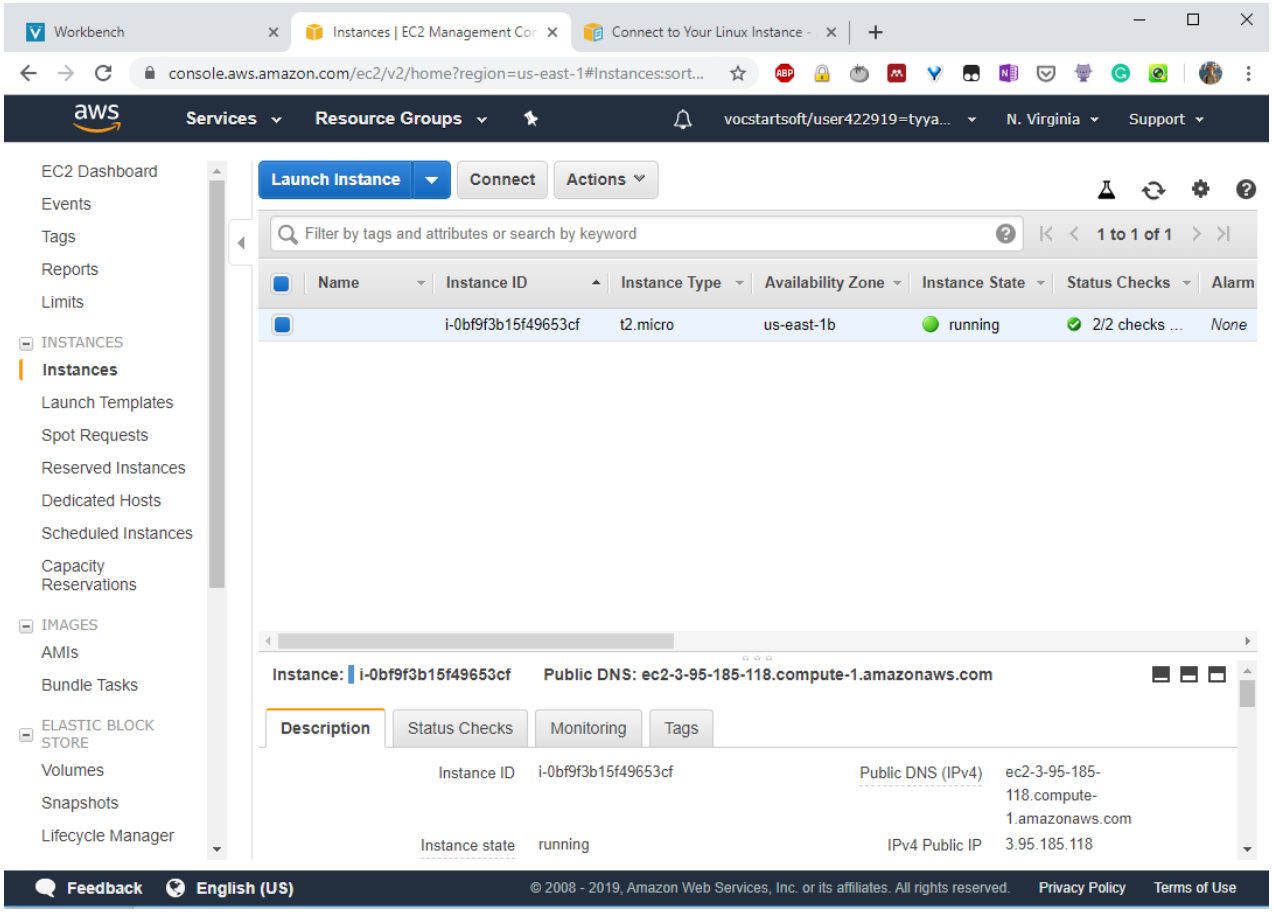
3. Create a new key pair (or select existing one) and download it.

Click Launch Instances and wait a few minutes until it is running.



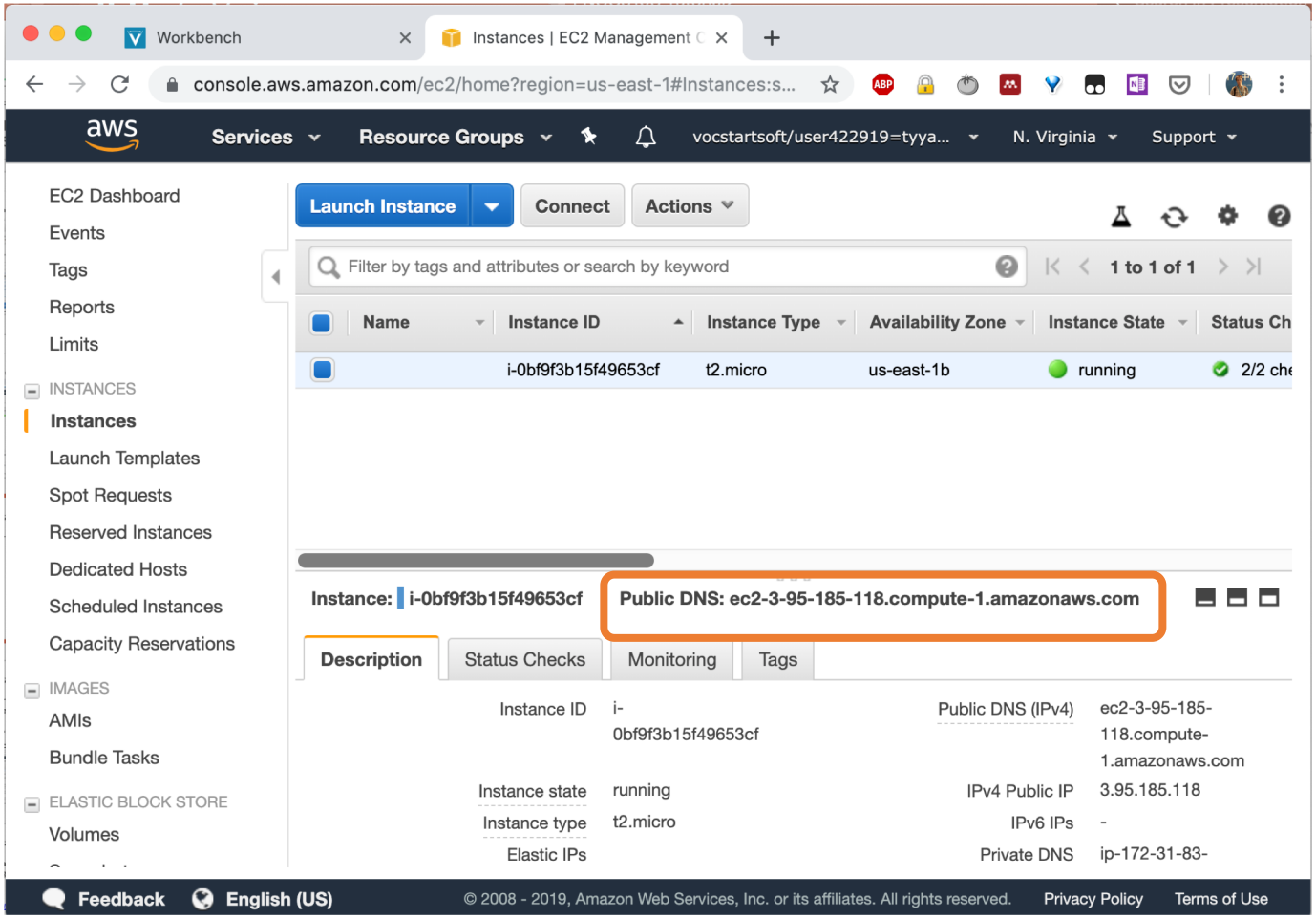
Create Micro Instance on EC2

4. In your EC2 management console you will see the state of your VM



Connect to the instance

- First, find the public DNS of your instance



Connect to the instance

- If you are in Linux or Windows 10, you can connect to your instance with ssh

```
 yangty@Terrill-MacBook  ~  cd Downloads/aws-tut
 yangty@Terrill-MacBook  ~/Downloads/aws-tut  ls
hadoop-aws.pem
 yangty@Terrill-MacBook  ~/Downloads/aws-tut  ssh -i hadoop-aws.pem ubuntu@ec2
-3-95-185-118.compute-1.amazonaws.com
```

- The key must be only readable to you
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstances.html

Connect to the instance

- If you are using Windows 8 or older version, you need an ssh client, putty is a great free ssh client.
- Follow the guide:
  - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html

Basic Linux Commands

- ls : list directory
- cd : change directory
- mkdir : create directory
- rm : remove file
- ps : show process status
- ssh: OpenSSH SSH client (remote login)
- wget: network downloader

Database Basics

- MySQL
- MONGODB

Installation and Connection of MySQL(Windows)

Download link:

https://www.mysql.com/

Switch to the downloaded folder:

```
cd C:\User\CSCI3100\Download\mysql-8.0.11\bin
```

Initialization(You will get a temporary password):

```
mysqld --initialize --console
```

Installation(Windows):

```
mysqld install
```

Start MySQL:

```
net start mysql
```

Connect MySQL:

```
mysql -h hostname -u username -p
```

Database Operation in MySQL

Create Database:

```
mysql> CREATE DATABASE CSCI3100;
```

Delete Database:

```
mysql> DROP DATABASE CSCI3100;
```

Select Database:

```
mysql> USE CSCI3100;
```

Table Operation in MySQL

Create Table:

```
mysql> CREATE TABLE student(
   -> student_id INT NOT NULL,
   -> name VARCHAR(100) NOT NULL,
   -> gender VARCHAR(40) NOT NULL,
   -> grade INT,
   -> PRIMARY KEY (student_id )
   -> )ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Delete Table:

```
mysql> DROP TABLE student;
```

Insert and Query Data in MySQL

Insert Data:

```
mysql> INSERT INTO student
    -> (student_id, name, gender, grade)
    -> VALUES
    -> (1155139458, ”MIKE", “MALE", 98);
```

Query Data:

```
mysql> SELECT * from student;
+--------------+-----------+-------------+-------+
|  student_id  |    name   |    gender   | grade |
+--------------+-----------+-------------+-------+
| 1155139458 |    MIKE   |    MALE     |   98  |
| 1155139462 |    BOB    |    MALE     |   89  |
| 1155139478 |    JANE   |  FEMALE     |   89  |
+--------------+-----------+-------------+-------+
3 rows in set (0.01 sec)
```

Common Clauses in MySQL

WHERE:

```
mysql> SELECT * from student WHERE grade<90;
+---------------+------------+--------------+-------+
|   student_id  |     name   |    gender    | grade |
+---------------+------------+--------------+-------+
| 1155139462 |     BOB     |    MALE     |    89   |
| 1155139478 |    JANE    |  FEMALE   |    89   |
+---------------+-------------+--------------+-------+
2 rows in set (0.01 sec)
```

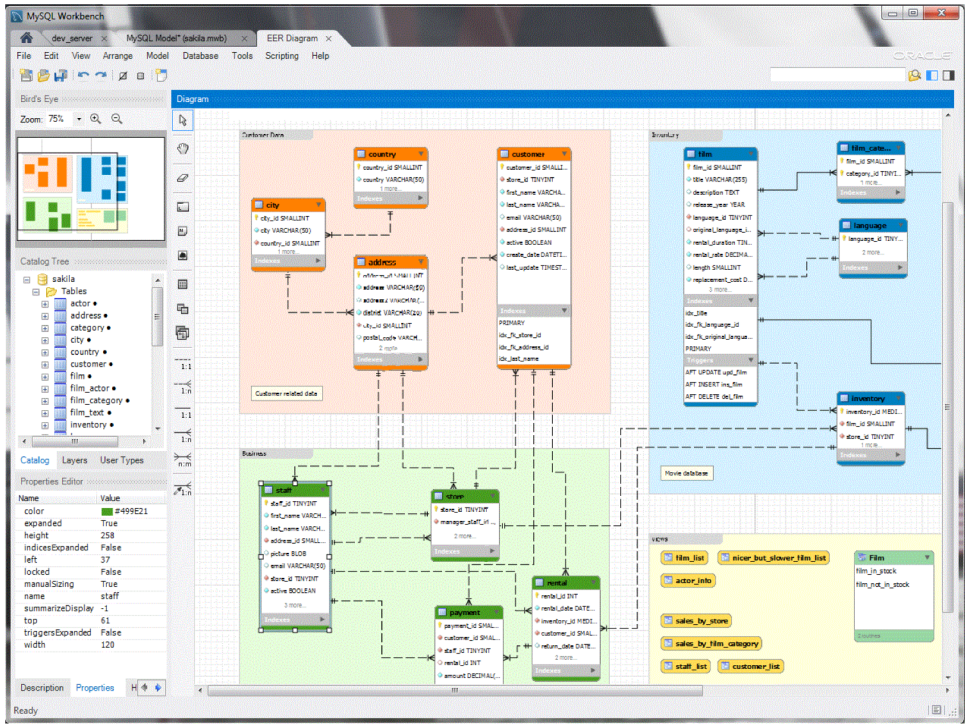
UPDATE:

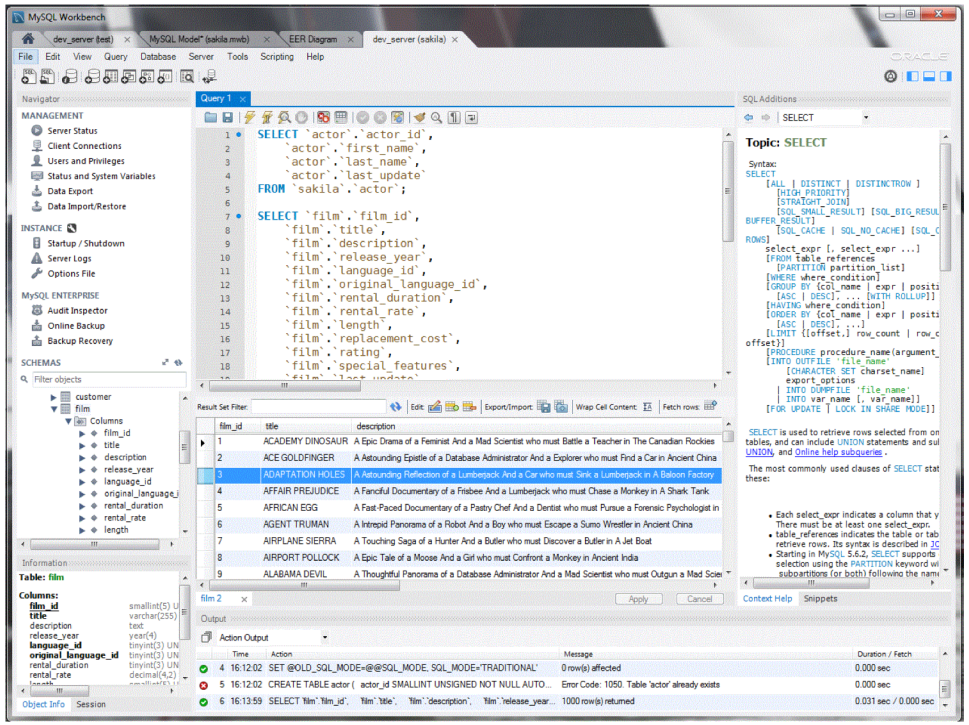
```
mysql> UPDATE student SET grade=85 WHERE student_id= 1155139458;
```

DELETE:

```
mysql> DELETE FROM student WHERE student_id= 1155139458;
```

MySQL WorkBench (GUI for MySQL)





SQL vs NoSQL Database

- SQL databases are known as relational databases, and have a **table-based** data structure, with a **strict, predefined** schema required.

- NoSQL databases, or non-relational databases, can be document based, graph databases, key-value pairs, or wide-column stores. NoSQL databases **don't require any predefined schema**, allowing you to work more freely with "unstructured data."

SQL vs NoSQL Database

- Relational databases are **vertically scalable**, but usually **more expensive**, whereas the horizontal scaling nature of NoSQL databases is more **cost efficient**.

- Since NoSQL has looser restrictions, it can even be processed in memory, e.g. Redis

MongoDB

Advantage:

- Weak consistency, ensure the access speed
- Easier to retrieve data because its storage method
- Rich third party support
- Good performance

What makes it different from MySQL?

MongoDB	RDBMS (E.g. MySQL)
Document-oriented and non-relational database	Relational database
Document based	Row based
Field based	Column based
Collection based and key–value pair	Table based
Gives JavaScript client for querying	Doesn't give JavaScript for querying
Has dynamic schema and ideal for hierarchical data storage	Has predefined schema and not good for hierarchical data storage
100 times faster and horizontally scalable through sharding	By increasing RAM, vertical scaling can happen

Installation and connection of MongoDB(Windows)

Download Link:

https://www.mongodb.com/

Create the data folder:

```
c:\>mkdir data
c:\>cd data
c:\data>mkdir db
```

Run MongoDB server:

```
C:\mongodb\bin\mongod --dbpath c:\data\db
```

Connect to MongoDB:

```
C:\mongodb\bin\mongo.exe
```

Start MongoDB service:

```
net start MongoDB
```

Connection URL to MongoDB service:

```
mongodb://admin:123456@localhost/
```

Conceptual Analogy between MySQL and MongoDB

MySQL	MongoDB
database	database
table	collection
row	document
column	field
index	index
table joins	
primary key	primary key

Database Operation in MongoDB

Create or select the database:

```
>use CSCI3100
```

Delete current database:

```
>db.dropDatabase()
```

Collection Operation in MongoDB

Create the collection:

```
db.createCollection(student, options)
```

Option	Type	Description
capped	bool	If the size of the collection is fixed
autoIndexId	bool	If create the auto index
size	int	The size of the collection
max	int	The max size of the document in the collection

Delete the collection:

```
db.student.drop()
```

Document Operation in MongoDB

Insert the document:

```
>db.student.insert({student_id: 1155139458,
    name: ‘MIKE',
    gender: 'MALE',
    grade: 98
})
```

Update the document:

```
>db.student.update({'student_id': 1155139458},{$set:{'grade' : 85}})
```

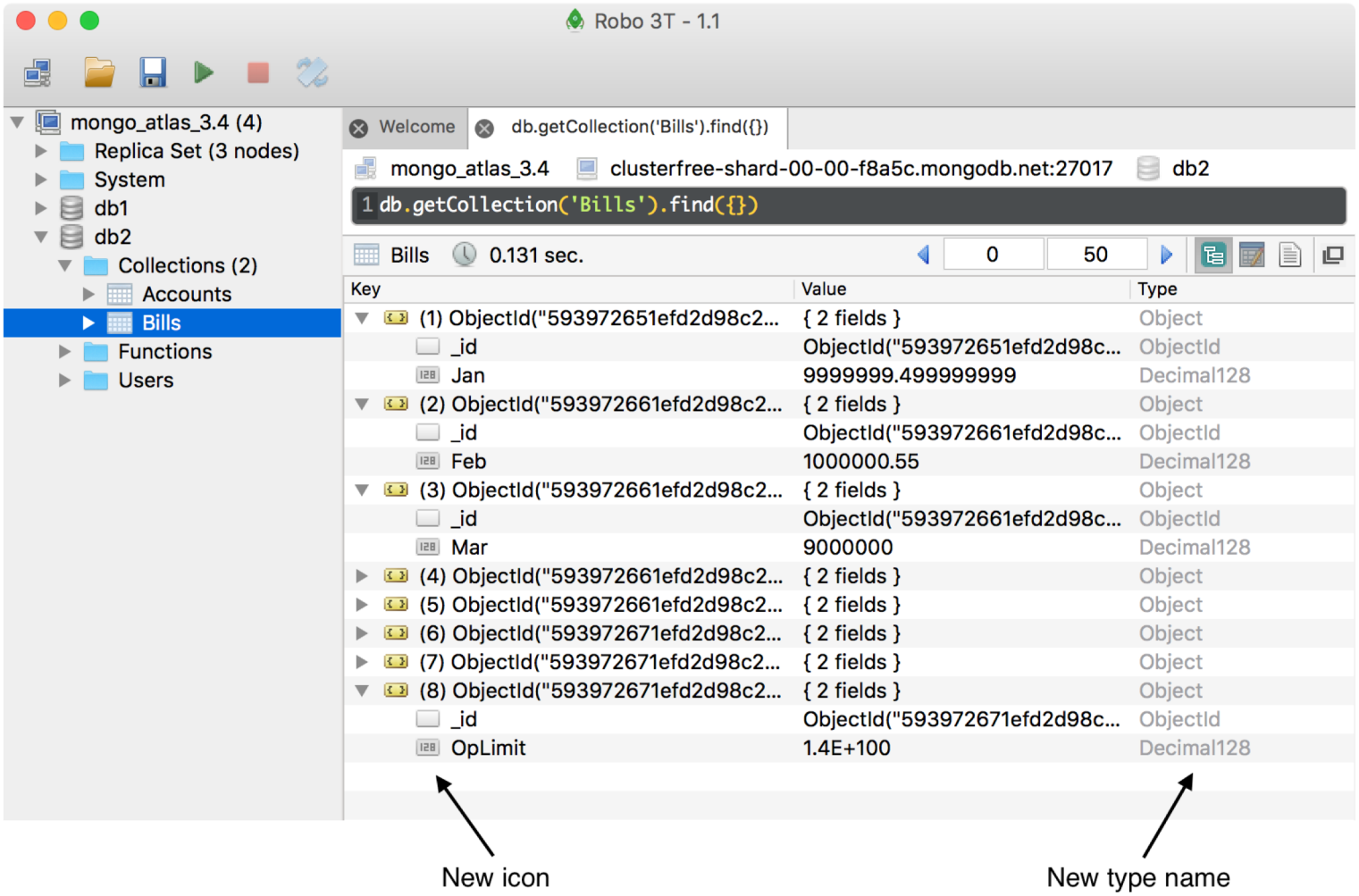
Delete the document:

```
>db.student.remove({'student_id': 1155139458})
```

Query the document:

```
> db. student.find({'student_id': 1155139458}).pretty()
```

Robo 3T (GUI for MongoDB)



Database Basics

- DATABSE Operation in NODE.JS

Connect to MySQL with node.js

```
var mysql = require('mysql');
var connection = mysql.createConnection({
    host: 'localhost',
    user: 'username',
    password: 'password',
    database: 'CSCI3100'
});
connection.connect();
```

Query Data in MySQL with node.js

```
// connect to mysql first
var sql = 'SELECT * FROM student';
connection.query(sql, function (err, result) {
    if (err) {
        console.log('[SELECT ERROR] - ', err.message);
        return;
    }
    console.log(result);
});
connection.end();
```

Insert Data in MySQL with node.js

```
//connect to mysql at first
var addSql = 'INSERT INTO
student(student_id,name,gender,grade) VALUES(?,?,?,?)';
var addSqlParams = [1155139468, 'MIKE', 'MALE', 98];
connection.query(addSql, addSqlParams, function (err, result)
{
    if (err) {
        console.log('[INSERT ERROR] - ', err.message);
        return;
    }
    console.log(result);
});
connection.end();
```

Update Data in MySQL with node.js

```
//connect to mysql at first
var modSql = 'UPDATE student SET grade = ? WHERE student_id =
?';
var modSqlParams = [85, 1155139458];
connection.query(modSql, modSqlParams, function (err, result)
{
    if (err) {
        console.log('[UPDATE ERROR] - ', err.message);
        return;
    }
    console.log(result.affectedRows);
});
connection.end();
```

Delete Data in MySQL with node.js

```
//connect to mysql at first
var delSql = 'DELETE FROM student where
student_id=1155139458';
connection.query(delSql, function (err, result) {
    if (err) {
        console.log('[DELETE ERROR] - ', err.message);
        return;
    }
    console.log(result.affectedRows);
});
connection.end();
```

Connect to MongoDB with node.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://admin:123456@localhost/";


MongoClient.connect(url, { useUnifiedTopology: true },
function (err, db) {
    if (err) {
        console.log('[CONNECT ERROR] - ', err.message);
        return;
    }
    console.log("Connection is successful!");
    db.close();
});
```

Query document in MongoDB with node.js

```
//set connection parameter at first
MongoClient.connect(url, { useUnifiedTopology: true }, function (err, db) {
    if (err) {
        console.log('[CONNECT ERROR] - ', err.message);
        return;
    }
    var dbo = db.db("CSCI3100");
    dbo.collection("student").find({ 'student_id': 1155139458
}).toArray(function (err, result) {
        if (err) {
            console.log('[QUERY ERROR] - ', err.message);
            return;
        }
        console.log(result);
        db.close();
    });
});
```

Insert document in MongoDB with node.js

```
//set connection parameter at first
MongoClient.connect(url, { useUnifiedTopology: true }, function (err, db) {
    if (err) {
        console.log('[CONNECT ERROR] - ', err.message);
        return;
    }
    var dbo = db.db("CSCI3100");
    var myobj = { student_id: 1155139458, name: 'MIKE', gender: 'MALE', grade: 98 };
    dbo.collection("student").insertOne(myobj, function (err, res) {
        if (err) {
            console.log('[INSERT ERROR] - ', err.message);
            return;
        }
        console.log(res);
        db.close();
    });
});
```

Update document in MongoDB with node.js

```
//set connection parameter at first
MongoClient.connect(url, { useUnifiedTopology: true }, function (err, db) {
    if (err) {
        console.log('[CONNECT ERROR] - ', err.message);
        return;
    }
    var dbo = db.db("CSCI3100");
    var whereStr = { "student_id": 1155139458 };
    var updateStr = { $set: { "grade": 85 } };
    dbo.collection("student").update(whereStr, updateStr, function (err, res) {
        if (err) {
            console.log('[UPDATE ERROR] - ', err.message);
            return;
        }
        console.log(res);
        db.close();
    });
});
```

Delete document in MongoDB with node.js

```
//set connection parameter at first
MongoClient.connect(url, { useUnifiedTopology: true }, function (err, db) {
    if (err) {
        console.log('[CONNECT ERROR] - ', err.message);
        return;
    }
    var dbo = db.db("CSCI3100");
    var whereStr = { "student_id": 1155139488 };
    dbo.collection("student").deleteOne(whereStr, function (err, obj) {
        if (err) {
            console.log('[DELETE ERROR] - ', err.message);
            return;
        }
        console.log(obj);
        db.close();
    });
});
```

References

- TOP 20 BEST NODEJS FRAMEWORKS FOR DEVELOPERS IN 2020
- Comparison of web frameworks
- Node.js vs. Django: Is JavaScript Better Than Python?
- Node.js Documents
- The SQL vs NoSQL Difference: MySQL vs MongoDB
- httpbin.org (https://httpbin.org/)
- Node.js Tutorial W3school
- How Browsers Work: Behind the scenes of modern web browsers

References

- Linux Command Line Cheat Sheet
- GNU/Linux Command-Line Tools Summary
- MySQL Tutorial for Beginners
- MongoDB Tutorials - The Official MongoDB Document (Recommended)
- MongoDB In 30 Minutes
- MySQL vs MongoDB? Which database is better?
- An Enterprise Architect's View - When to Use MongoDB
- MySQL WorkBench (GUI for MySQL)
- Robo3T (GUI for MongoDB)

Thanks!