

SCHOOL OF COMPUTING

ST0505

ESDE (Cloud Computing)

CA2 REPORT

AY2021S2 2A22

By

Ong Jingjie Cleavon (P1935868)

Muhammad Kabir Bin A Wahab (P1936111)

- 1. Diagram**
- 2. Code snippets w/ Documentation**
- 3. Screenshot of testing result**
- 4. Troubleshooting**

EC2 S3 RDS

Create Instance

1. On the AWS Console, go to Services
2. Select EC2, and select launch instance
3. In step 1, choose the community AMI: Ubuntu Server 18.04 LTS (HVM), SSD Volume Type 64-bit (x86)
4. In step 6, Choose a Security Group, change source to 'My IP', click 'review and launch' button, then select 'launch' button.
5. Create a new keypair and save it on your computer as 'WPFOC-KEYPAIR'
6. Launch Instance

Connecting to the Created Remote Instance Using Putty

1. Install putty (PuttyGen will be installed along with it)
2. Open PuttyGen
3. Load Private Key. Under Type of key to generate, choose RSA

(Note: If you're using an older version of PuTTYgen, choose SSH-2 RSA)

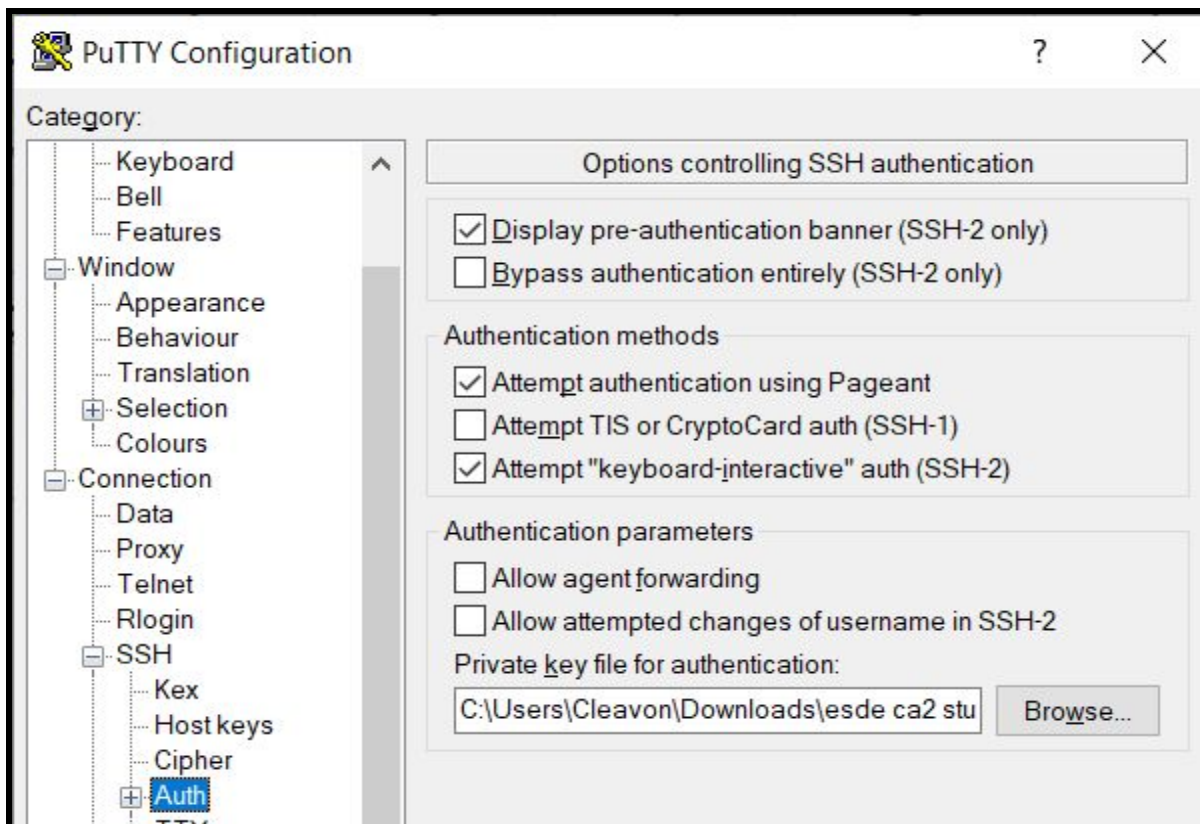
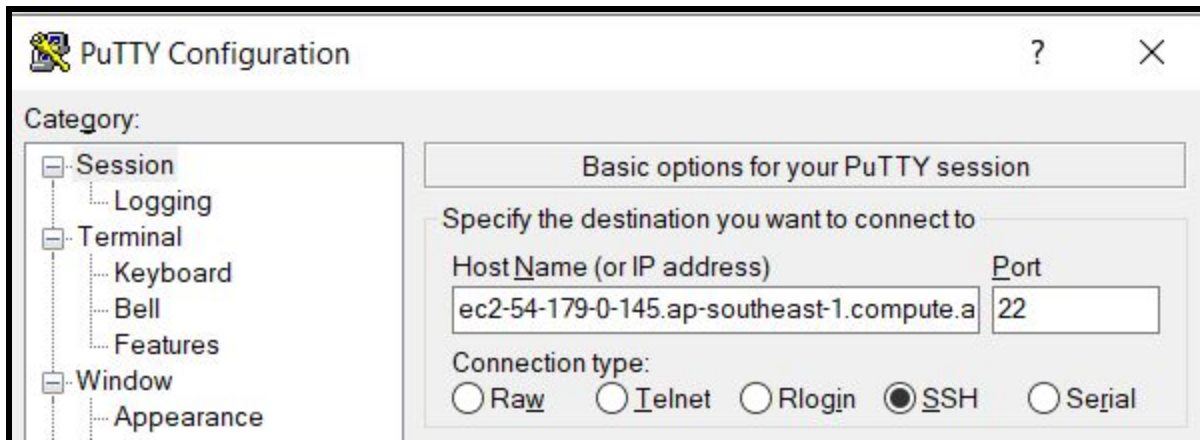
(Note: By default, PuTTYgen displays only files with the extension .ppk. To locate your .pem file, select the option to display files of all types.)

4. Select your .pem file for the key pair that you specified when you launched your instance and choose Open. Choose OK.
5. It will generate the private key. Save private key to a file by selecting "Save private key", name it as "WPFOC.ppk"
6. Open Putty, Select the "Connection" - SSH - Auth and then load the .ppk file.
7. Fill the Hostname field with <your public dns>. Also you can save the session by giving it a name and clicking on "Save".

9. Click on Open, and “yes” to login as “ubuntu” to the server.

Here are the screenshots for reference:

Name	Date modified	Type	Size
putty-64bit-0.74-installer	31/1/2021 2:23 AM	Windows Installer ...	2,777 KB
WPFOC	31/1/2021 2:44 AM	PuTTY Private Key ...	2 KB
WPFOC-KEYPAIR.pem	31/1/2021 1:53 AM	PEM File	2 KB



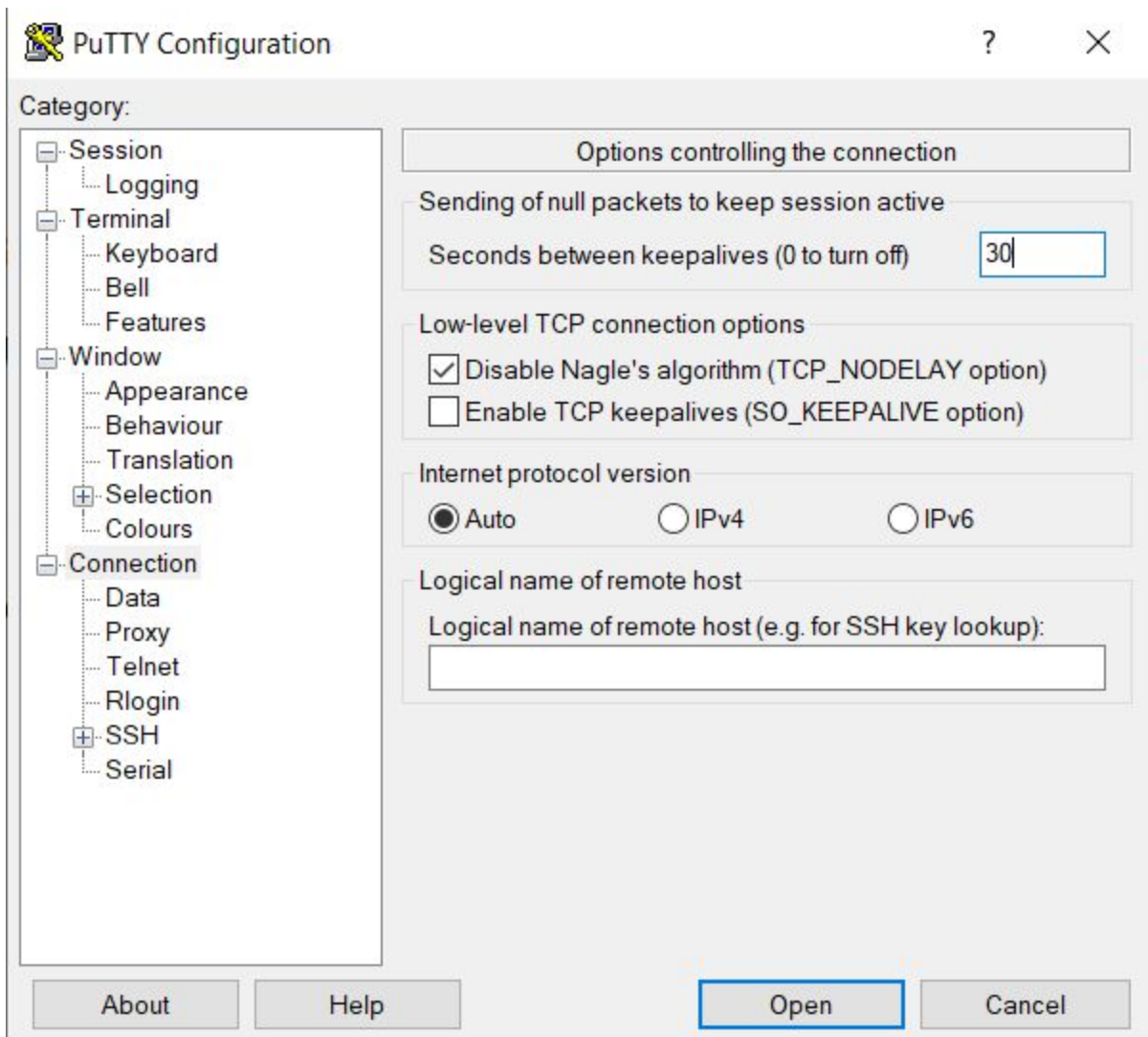
```
ubuntu@ip-172-31-39-19: ~  
login as: ubuntu  
Authenticating with public key "imported-openssh-key"  
Passphrase for key "imported-openssh-key":  
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1029-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of Thu Feb  4 07:50:54 UTC 2021  
  
System load:  0.0           Processes:            103  
Usage of /:   27.1% of 7.69GB Users logged in:      1  
Memory usage: 37%          IP address for eth0: 172.31.39.19  
Swap usage:   0%  
  
* Introducing self-healing high availability clusters in MicroK8s.  
  Simple, hardened, Kubernetes for production, from RaspberryPi to DC.  
  
    https://microk8s.io/high-availability  
  
* Canonical Livepatch is available for installation.  
  - Reduce system reboots and improve kernel security. Activate at:  
    https://ubuntu.com/livepatch  
  
28 packages can be updated.  
0 updates are security updates.  
  
New release '20.04.1 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
*** System restart required ***  
Last login: Thu Feb  4 07:42:24 2021 from 129.126.58.245  
ubuntu@ip-172-31-39-19:~$
```

Once you have this, You are logged in now.

Troubleshoot

How to change Putty SSH connection to never timeout when the user is idle?

If you go to your putty settings -> Connection and set the value of "Seconds between keepalives" to 30 seconds.



Using Amazon S3 for your Code Repository

1. Right click on a folder you wish to work on and select the “Compressed (zipped) folder” option in “Send to”.
2. Login to your AWS console and select S3. Click on the Create bucket button, enter a unique name for your bucket and click on the Next button
3. Click the ‘Next’ button again and uncheck all the checkboxes in the ‘Set Permissions’ tab.
4. Click the ‘Next’ button and click the ‘Create bucket’ button.
5. Select the newly created bucket and upload the project’s zip file to the bucket.
6. Click on the zip file and then click on the Make Public button.
7. Record down the endpoint for your source code for later steps

Install Node on EC2 Instance

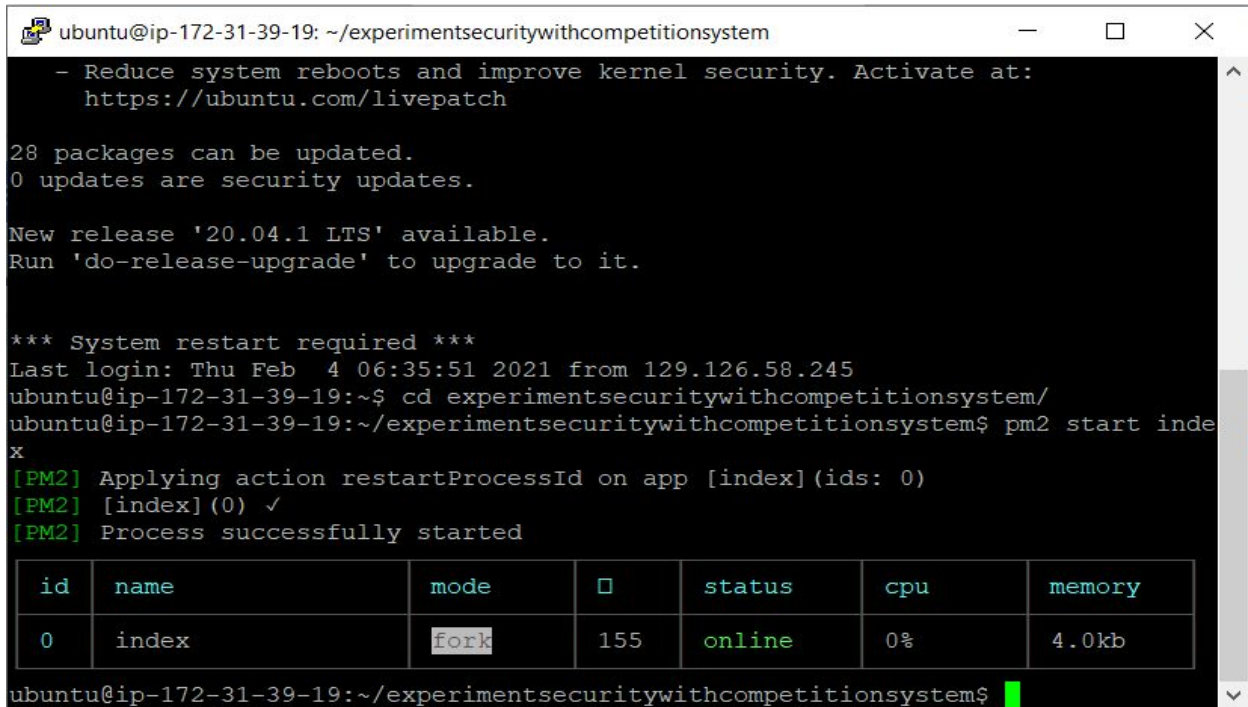
1. Install Node on the instance in your terminal or git bash by running this command: `curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.sh | bash`
2. Close and reopen your terminal to start using nvm
3. Use nvm to install Node.js because nvm can install multiple versions of Node.js and allow you to switch between them using this command: `~/.nvm/nvm.sh`
4. Installing node version 8.10.: `nvm install 8.10`
5. Check your node version: `node -v`

Copy code on your EC2 instance and install dependencies to your home directory(/home/ubuntu)

1. Get code from S3 repository that was previously noted down: wget <https://cc-backend.s3.amazonaws.com/experimentsecuritywithcompetitionsystem.zip>
2. Verify the node application is copied to your home directory: ls
3. Install unzip: sudo apt install unzip
4. Extract the zip file: unzip [experimentsecuritywithcompetitionsystem.zip](#)
5. Install Dependencies if your zip file does not contain all the packages in the node_modules: cd [experimentsecuritywithcompetitionsystem](#), then npm install

Start server to run forever

1. Install pm2: npm install -g pm2
2. Go to the directory and run: pm2 start index (or index.js)



The terminal window shows the following output:

```
ubuntu@ip-172-31-39-19: ~/experimentsecuritywithcompetitionsystem
- Reduce system reboots and improve kernel security. Activate at:
https://ubuntu.com/livepatch

28 packages can be updated.
0 updates are security updates.

New release '20.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Thu Feb  4 06:35:51 2021 from 129.126.58.245
ubuntu@ip-172-31-39-19:~$ cd experimentsecuritywithcompetitionsystem/
ubuntu@ip-172-31-39-19:~/experimentsecuritywithcompetitionsystem$ pm2 start index
[PM2] Applying action restartProcessId on app [index] (ids: 0)
[PM2] [index] (0) ✓
[PM2] Process successfully started
```

id	name	mode	pid	status	cpu	memory
0	index	fork	155	online	0%	4.0kb

```
ubuntu@ip-172-31-39-19:~/experimentsecuritywithcompetitionsystem$
```

3. Open in browser: [http://\[your public IP address or domain name\]:5000/](http://[your public IP address or domain name]:5000/)

Troubleshoot

-Error reaching site

Configure Security Group to ensure you open the port where your Application will run on

1. Go to Network Security - Security Group, and select "Create Security Group"
2. Add Security Group Name and Description
3. Add Inbound Rules
 - a. Custom Source for SSH : **0.0.0.0/0**
 - b. Custom TCP, Port Range: 5000, Custom Source : **0.0.0.0/0 & ::/0**
4. Once completed, go to Network & Security - Network Interfaces
5. Select instance which you want to change Security Group on - Actions - Change Security Group
6. Select your newly created group and click 'Save'

The screenshot displays the AWS Management Console interface. The top section shows a list of Security Groups. The second group, 'sg-0dcea3480abf000b1' (launch-wizard-1), is selected. Below this, the 'Inbound rules' tab is active, showing three rules: SSH (port 22), Custom TCP (port 5000), and another Custom TCP (port 5000). The source for all rules is set to 0.0.0.0/0 or ::/0.

Security Groups (1/3) [Info](#)

Filter security groups

Name	Security group ID	Security group name	VPC ID	Description	Owner
-	sg-0ba279cb65903da89	launch-wizard-32	vpc-eae8158c	Allows SSH access to d...	23547701595
<input checked="" type="checkbox"/>	sg-0dcea3480abf000b1	launch-wizard-1	vpc-eae8158c	launch-wizard-1 create...	23547701595
<input type="checkbox"/>	sg-6c27fd24	default	vpc-eae8158c	default VPC security gr...	23547701595

sg-0dcea3480abf000b1 - launch-wizard-1

Details **Inbound rules** Outbound rules Tags

Inbound rules [Edit inbound rules](#)

Type	Protocol	Port range	Source	Description - optional
SSH	TCP	22	0.0.0.0/0	-
Custom TCP	TCP	5000	0.0.0.0/0	-
Custom TCP	TCP	5000	::/0	-

Configure pm2

Run: **pm2 startup**

After running the command, you will get a command starting with “sudo”, copy it from sudo till end of the next line and paste it in the terminal and enter

- To freeze a process list via reboot: **\$ pm2 save**
- To restart server: **\$ pm2 restart server**
- To reload server: **\$ pm2 reload server**
- To stop server: **\$ pm2 stop server**
- To delete server: **\$ pm2 delete server**

Create a new Amazon Web Service RDS instance with Amazon RDS

1. On the AWS Console go to Services - Database - RDS
2. Click on 'Create database'
3. Keep everything as default, except the following:
 - a. Engine Options - Engine type - MySQL
 - b. Templates - Free Tier
 - c. Credentials Settings - Password
 - d. Additional Connectivity Configuration - Publicly Accessible (Yes)
4. Click on 'Create database', you should see your database being created
5. View information about the database by clicking on the name
6. Under Connectivity & Security, note down the **endpoint** and **port number**

Connect & send SQL queries remotely to an Amazon Web Service RDS instance using MySQL Workbench

1. Launch MySQL Workbench and locate the '+' icon to add a new connection
2. Fill up the form with the RDS database information that you had recorded down (**endpoint, port and username**)
3. Click on Test connection, you should be prompted a dialog asking for your MySQL password
4. Submit your password and you should finally have a notice, stating that the connection is working fine
5. You should be able to see a new MySQL connection created

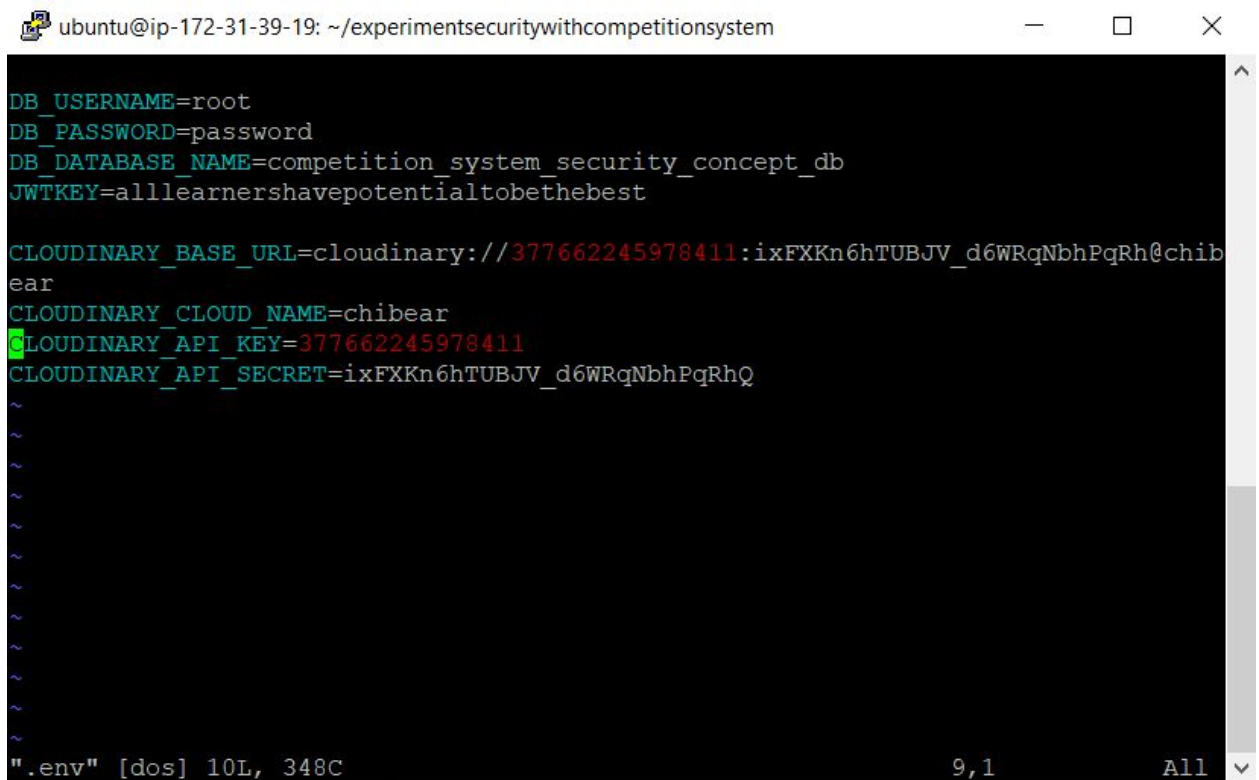
Troubleshoot

If there is a connection error, check the Security Group for the database if it allows your application server to connect.

1. Connectivity & security - VPC security group link
2. Select "Edit Inbound Rule"
3. Add rules:
 - a. All traffic
 - b. Custom Source **0.0.0.0/0 & ::/0**
 - c. Save Rules
4. Try connecting MySQL with AWS RDS again. The connection should be working now.

Load SQL text file into the new connection and configure database connection in EC2

1. Click on Administration - Data import/Restore to import the SQL Script file (Snapsell-v2) into the workbench
2. Execute it by clicking on the lightning icon
3. After the script file has been loaded, open up the Putty terminal and cd into the directory with the .env file
4. Edit .env by opening the file in the terminal: **vim .env**
5. Click on the character 'i' on the keyboard to start editing.



The screenshot shows a terminal window with the title bar "ubuntu@ip-172-31-39-19: ~/experimentsecuritywithcompetitionsystem". The terminal content displays the following environment variables:

```
DB_USERNAME=root
DB_PASSWORD=password
DB_DATABASE_NAME=competition_system_security_concept_db
JWTKEY=alllearnershavepotentialtobethebest

CLOUDINARY_BASE_URL=cloudinary://377662245978411:ixFXKn6hTUBJV_d6WRqNbhpqRh@chibear
CLOUDINARY_CLOUD_NAME=chibear
CLOUDINARY_API_KEY=377662245978411
CLOUDINARY_API_SECRET=ixFXKn6hTUBJV_d6WRqNbhpqRhQ
```

At the bottom of the terminal, the status bar shows: ".env" [dos] 10L, 348C, 9,1, and All.

6. Edit the following:
 - a. Change host to endpoint of the RDS
 - b. Change user to username set in RDS
 - c. Change password to password set in RDS
 - d. Change database name to table in MySQL
7. To save changes, press '**Esc**', type '**:wq**' and then press '**Enter**'

8. Restart the pm2 server: **pm2 restart server**
9. Open in browser and test for functionality by using any api:

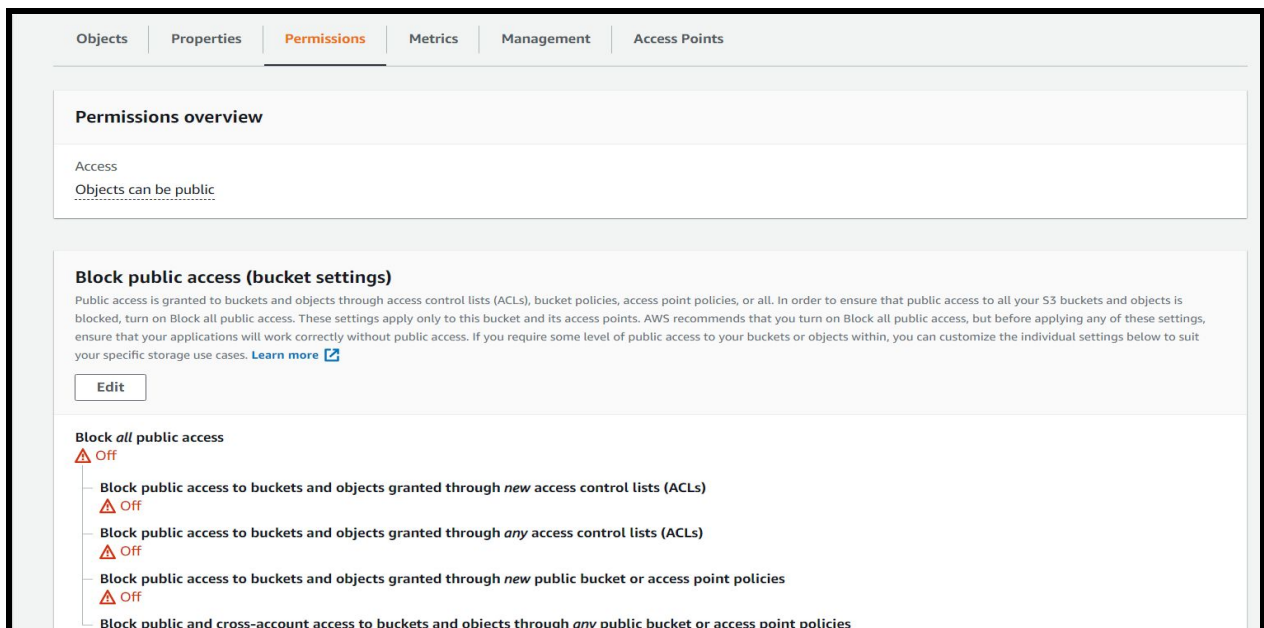
http://[your public IP address or domain name]:5000/api/user/design/100

Data successfully retrieved from database:

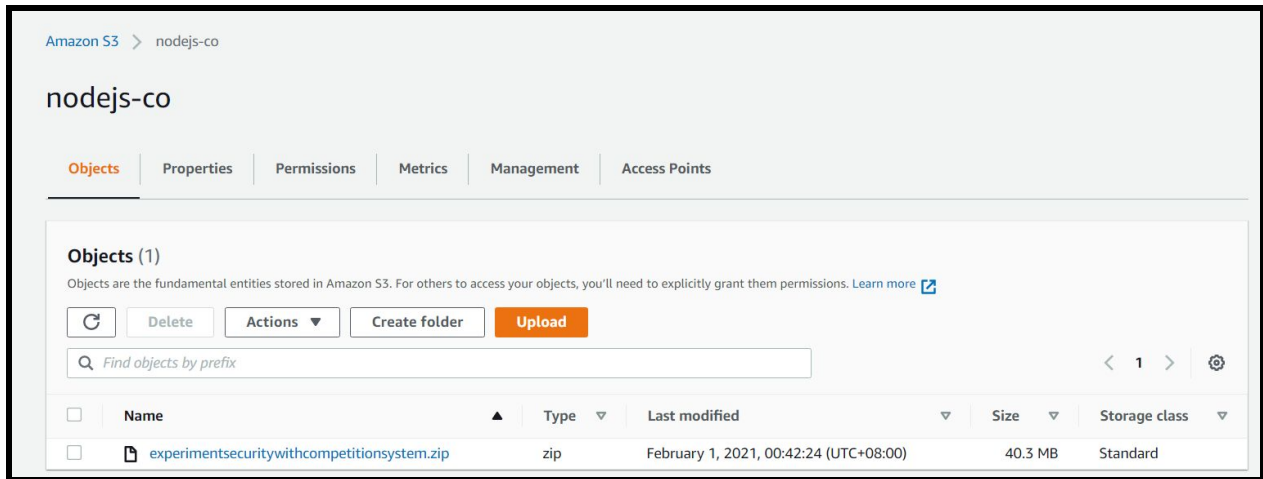


Host a website - Create a bucket

1. Under services, click on S3
2. Click on create bucket
3. Enter the Bucket name (for example, example.com)
4. Click the Next button again and uncheck all the checkboxes in the Set Permissions tab.



5. Click the Next button and then click the Create bucket button.



Enable static website hosting

1. In the Bucket name list, choose the bucket that you want to use for your static website.
2. Choose Properties.
3. Choose Static website hosting.
4. Choose Use this bucket to host a website.
5. Enter the name of your index document.
6. The index document name is typically index.html. The index document name is case sensitive and must exactly match the file name of the HTML index document that you plan to upload to your S3 bucket.
7. Under Static website hosting, note the Endpoint
8. Click on save

Add a bucket policy that makes your bucket content publicly available

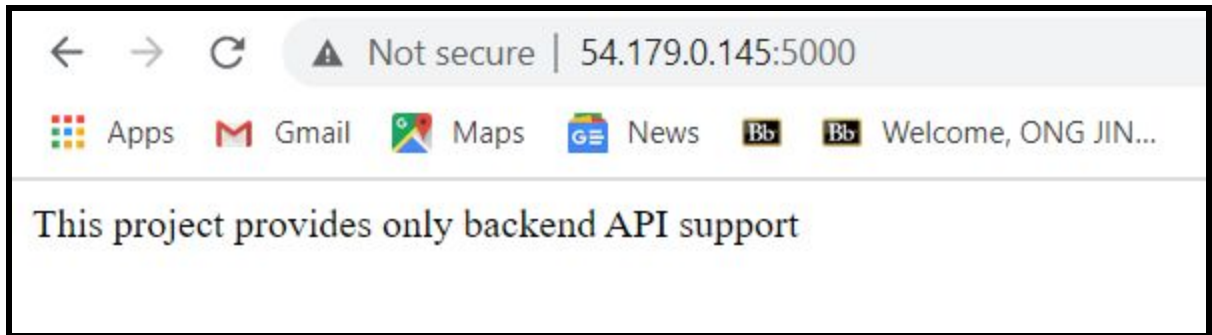
1. Under Buckets, select the name of your bucket.
2. Select Permissions.
3. Select Bucket Policy.
4. To grant public read access for your website, copy the following bucket policy, and paste it in the Bucket policy editor.

```
{  
  
  "Version": "2012-10-17",  
  
  "Statement": [  
  
    {  
  
      "Sid": "PublicReadGetObject",  
  
      "Effect": "Allow",  
  
      "Principal": "*",  
  
      "Action": [  
  
        "s3:GetObject"  
  
      ],  
  
      "Resource": [  
  
        "arn:aws:s3:::example.com/*"  
  
      ]  
  
    }  
  
  ]  
  
}
```

5. Update the Resource to include your bucket name.
6. Click on save

Test your website endpoint

1. To test your website, enter the website endpoint in your browser that you previously noted down. If your browser displays your .html page, the website was successfully deployed.



Diagram

S3	Amazon S3 (Short for Amazon Simple Storage Service) has a simple web services interface that we can use to store and retrieve data from anywhere on the web at any time.
EC2	Amazon EC2 (Short for Amazon Elastic Compute Cloud) allows the user to develop and deploy applications at a much faster rate. It can also create "instances" which are virtual computing environments
RDS	It makes it easier for the user to scale databases in the cloud.

Part 2: API Gateway

Code Snippets and Inline Documentation

Launch Cloud9, create a new environment. Push the files from S3 bucket, and update the necessary files. Create a “create_bucket.js” file and enter the following code.

```
var

    AWS = require("aws-sdk"),

    S3API = new AWS.S3({

        apiVersion: "2006-03-01",

        region: "us-east-1"

    });

(function createBucket(){

    var

    params = {

        Bucket: "nodejs-kw", //bucket name

    };

    S3API.createBucket(params, function(error, data){

        console.log(error, data);

    });

})();
```

Create a new file called “public_policy.json” here is the code that I have used. This is to create a policy that allows anyone to read from the bucket.

```
{  
  
  "Version": "2012-10-17",  
  
  "Statement": [{  
  
    "Sid": "PublicReadGetObject",  
  
    "Effect": "Allow",  
  
    "Principal": "*",  
  
    "Action": ["s3:GetObject"],  
  
    "Resource": ["arn:aws:s3:::nodejs-co"  
  
    ]  
  
  }  
  
]  
}
```

Create a file name “upload_items.js” and upload the following code inside.

```
var  
  
  AWS = require("aws-sdk"),  
  
  S3API = new AWS.S3({  
  
    apiVersion: "2006-03-01",  
  
    region: "us-east-1"  
  
  }),  
  
  FS = require("fs"),  
  
  bucket_name_str = "nodejs-co";
```

```

function uploadItemAsBinary(key_name_str, content_type_str, bin){

    var params = {

        Bucket: bucket_name_str,

        Key: key_name_str,

        Body: bin,

        ContentType: content_type_str,

        CacheControl: "max-age=0"

    };

    S3API.putObject(params, function(error, data){

        console.log(error, data);

    });

}

(function init(){

    var

    file_path_str = "./",

    file_name_str = "experimentsecuritywithcompetitionsystem.zip",

    config_bin = FS.readFileSync(file_path_str + file_name_str);

    uploadItemAsBinary(file_name_str, "application/zip", config_bin);

})();

```

Ensure that my table is created by going to dynamoDB and checking

Add the following code to a file called query.js and run it.

```
exports.handler = function(event, context, callback){

    var

        AWS = require("aws-sdk"),

        DDB = new AWS.DynamoDB({

            apiVersion: "2012-08-10",

            region: "us-east-1"

        });

    function queryIndex(title_str, cb){

        var

            params = {

                ExpressionAttributeValues: {

                    ":title": {

                        S: title_str

                    }

                },

                KeyConditionExpression: "title = :title",

                TableName: "file",

                IndexName: "file_id"

            };

        DDB.query(params, function(err, data){

            var

                cat_reply_arr = [];

            if(err){
```

```

        throw err;

    }

    if(data.Items.length === 0){

        return cb(null, []);

    }

    for(var i_int = 0; i_int < data.Items.length; i_int += 1){

        cat_reply_arr.push(data.Items[i_int]);

    }

    cb(null, cat_reply_arr);

});

}

function scanTable(cb){

    var

    params = {

        TableName: "file"

    };

    DDB.scan(params, function(err, data){

        if(err){

            throw err;

        }

        cb(null, data.Items);

    });

}

```

```
(function init(){  
  
    var  
  
    title_str = "all",  
  
    cb = null;  
  
    if(process.argv[2] !== undefined){  
  
        console.log("Local test for " + process.argv[2]);  
  
        title_str = process.argv[2];  
  
        cb = console.log;  
  
    }else{  
  
        console.log("Running in lambda");  
  
        console.log(event);  
  
        cb = callback; //becomes available in lambda  
  
        title_str = event.title_str;  
  
    }  
  
    if(title_str === "All"){  
  
        scanTable(cb);  
  
    }else{  
  
        queryIndex(title_str, cb);  
  
    }  
  
    }) ();  
};
```

Create a new API

Add this code into existing config.js file

```
var API_ENDPOINT_STR =  
"https://w94f4tzt2m.execute-api.us-east-1.amazonaws.co";
```

Enter this string of code in seeddata.js

```
var  
  
    AWS = require("aws-sdk"),  
  
    DDB = new AWS.DynamoDB({  
  
        apiVersion: "2012-08-10",  
  
        region: "us-east-1"  
  
    }),  
  
    DATA_ARR = require("../data.json");  
  
function addNewItemFromJSON() {  
  
    var  
  
        data = {},  
  
        data_formatted_arr = [],  
  
        params = {};  
  
  
  
    for(var i_int = 0; i_int < DATA_ARR.length; i_int += 1){  
  
        data = {  
  
            PutRequest: {
```



```
Item: {

    id: {

        "N": DATA_ARR[i_int]._file_id_int

    },

    cloudinaryid: {

        "S": DATA_ARR[i_int].cloudinary_file_id_str

    },

    cloudinaryurl: {

        "S": DATA_ARR[i_int].cloudinary_url_str

    },

    designtitle: {

        "N": DATA_ARR[i_int].design_title_str

    },

    created_by: {

        "S": DATA_ARR[i_int].created_by_str

    },

    image: {

        "S": DATA_ARR[i_int].image_str

    }

}

};

data_formatted_arr.push(data);
```

```

    }

    params = {

    RequestItems: {

        "file": data_formatted_arr.reverse()

    }

    };

    DDB.batchWriteItem(params, function(err, data){

    if(err){

        throw err;

    }

    console.log("OK");

    });

}

function init(){

    addNewItemsFromJSON();

}

init();

```

Next is my code in create_lambda.js

```

var

    AWS = require("aws-sdk"),

```

```
LAMBDA = new AWS.Lambda({

  apiVersion: "2015-03-31",

  region: "us-east-1"

});

function createLambdaFromZip() {

  var

  params = {

    Code: {

      S3Bucket: "nodejs-kw",

      S3Key: "experimentsecuritywithcompetitionsystem.zip"

    },

    Description: "competition_system_security_concept_db",

    FunctionName: "getOneDesignData",

    Handler: "query.handler",

    MemorySize: 128,

    Publish: true,

    Role: "arn:aws:iam::278011394123:role/lambda-role",

    Runtime: "nodejs12.x",

    Timeout: 30,

  };

  LAMBDA.createFunction(params, function(err, data){
```

```

        console.log(err, data);

    });

}

(function init(){

    createLambdaFromZip();

})();

```

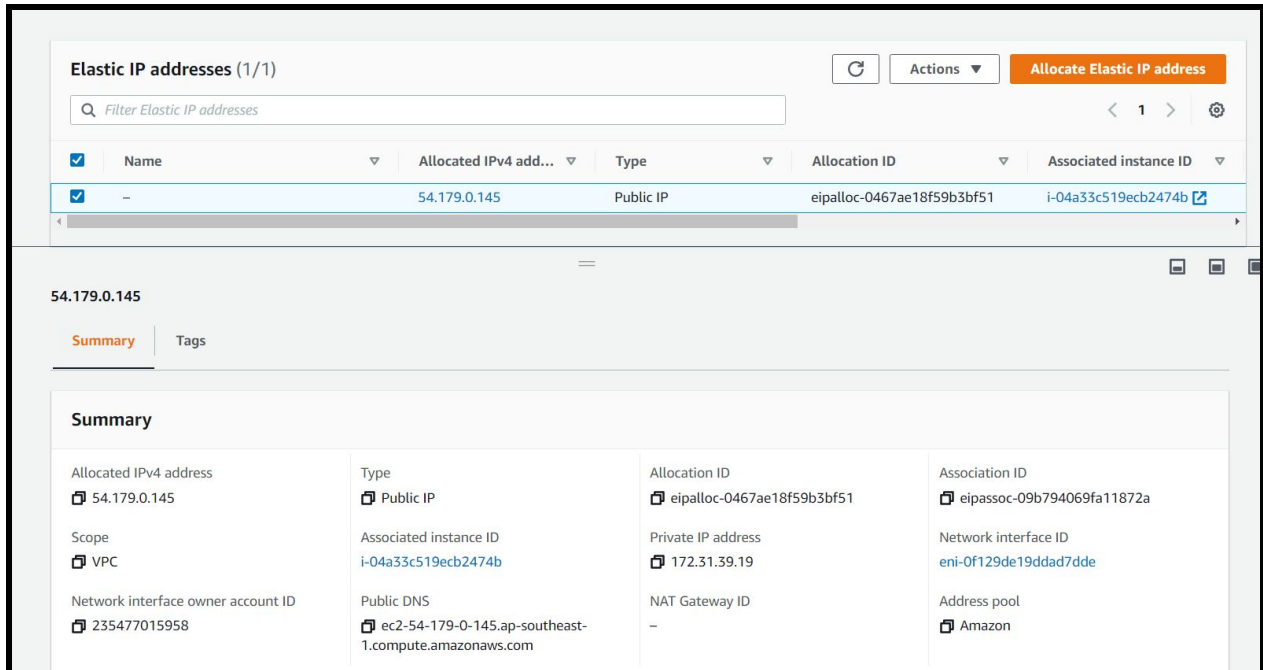
Troubleshoot

1. Problem 1:

Did not know why I couldn't get the backend in the EC2 to work

1.1. Solution:

Making use of the Elastic IP



Elastic IP addresses (1/1)

Filter Elastic IP addresses

<input checked="" type="checkbox"/>	Name	Allocated IPv4 address	Type	Allocation ID	Associated instance ID
<input checked="" type="checkbox"/>	-	54.179.0.145	Public IP	eipalloc-0467ae18f59b3bf51	i-04a33c519ecb2474b

54.179.0.145

Summary | Tags

Summary			
Allocated IPv4 address 54.179.0.145	Type Public IP	Allocation ID eipalloc-0467ae18f59b3bf51	Association ID eipassoc-09b794069fa11872a
Scope VPC	Associated instance ID i-04a33c519ecb2474b	Private IP address 172.31.39.19	Network interface ID eni-0f129de19ddad7dde
Network interface owner account ID 235477015958	Public DNS ec2-54-179-0-145.ap-southeast-1.compute.amazonaws.com	NAT Gateway ID -	Address pool Amazon

2. Index.js keeps having issues

```
/home/ubuntu/.pm2/logs/index-out.log last 15 lines:
/home/ubuntu/.pm2/logs/index-error.log last 15 lines:
0|index | at tryModuleLoad (module.js:505:12)
0|index | at Function.Module._load (module.js:497:3)
0|index | at Module.require (module.js:596:17)
0|index | node-pre-gyp info This Node instance does not support builds for N-API version 3
0|index | Error: The N-API version of this Node instance is 1. This module supports N-API version(s) 3. This Node instance cannot run this module.
0|index | at Object.module.exports.validate_package_json (/home/ubuntu/experimentsecuritywithcompetitionsystem/node_modules/node-pre-gyp/lib/util/napi.js:82:9)
0|index | at Object.validate_config (/home/ubuntu/experimentsecuritywithcompetitionsystem/node_modules/node-pre-gyp/lib/util/versioning.js:229:10)
0|index | at Object.exports.find (/home/ubuntu/experimentsecuritywithcompetitionsystem/node_modules/node-pre-gyp/lib/pre-binding.js:21:15)
0|index | at Object.<anonymous> (/home/ubuntu/experimentsecuritywithcompetitionsystem/node_modules/bcrypt/bcrypt.js:5:27)
0|index | at Module._compile (module.js:652:30)
0|index | at Object.Module._extensions..js (module.js:663:10)
0|index | at Module.load (module.js:565:32)
0|index | at tryModuleLoad (module.js:505:12)
0|index | at Function.Module._load (module.js:497:3)
0|index | at Module.require (module.js:596:17)
```

2.1 Solution:

Install bcryptjs.