



# Cairn Alumni Portal

*Cairn University's School of Business  
Software Engineering*

Miguel Augusto Tapia  
*Tapia.Tech Owner*

Mr. Worth  
*C.University Representative*

**Project Implementation Plan**

April 15, 2020

# C.U Connect

## Overview:

In order to deploy the application, there are 3 main tiers which go as follows: **application, database, and cloud provider**. First, the application tier focuses on changes that need to be made to the existing codebase. Next, the database tier addresses necessary changes to the database solution. Finally, the cloud provider section ties everything together and creates a bridge between everything. If developers successfully replicate the following steps, the application will work as expected on the cloud. The current version is online at the following link: <https://cu-connect.herokuapp.com/>.

## Application Tier

At the application level, there are a couple of things that must take place for implementation and deployment. First, the development team will install the following packages for security and speed optimizations: Helmet.js, compression, and morgan. These packages execute as middlewares. The morgan package, which has a will create an access.log file that has information about the user utilizing the application.

```
// Function Access And Write To Log File
const accessLogStream = fs.createWriteStream(
  path.join(__dirname, "access.log"),
  { flags: "a" }
);

// Filter Through Requests For Compression, Security, And Logging
app.use(helmet());
app.use(compression());
app.use(morgan("combined", { stream: accessLogStream }));
```

Once this takes place, the developers will shield all secure, sensitive information, like database users, passwords, keys, hashes, etc, through process.env and give the cloud provider access to the application port. The following shows the implementation for the database URI, session secret, and process control to the cloud provider.

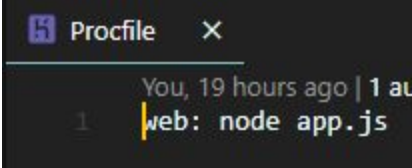
```
// Declare Mongo URI
const MONGODB_URI = `mongodb+srv://${process.env.MONGO_USER}:${process.env.MONGO_PASSWORD}@cu-connect-main-eipxr.mongodb.net/${process.env.MONGO_DEFAULT_DATABASE}`;
```

```
app.use(
  session({
    secret: process.env.SESSION_SECRET,
    resave: false,
    saveUninitialized: false,
    store: store
  })
);
```

```
// Initialize Express Server & Port
const app = express();
const port = process.env.PORT || 3000;
```

Finally, the implementor must add engines (node: 12.13.0) to the package.json file and ensure there is a start script with the following: “node app.js.” Additionally, he must create a procfile telling heroku (cloud provider) what will run.

```
"engines": {
  "node": "12.13.0"
},
"scripts": {
  "start": "node app.js",
  "start:dev": "nodemon app.js",
  "test": "echo \"Error: no test specified\" && exit 1"
},
```



The screenshot shows a Heroku Procfile window with the title 'Procfile' and a close button. The content of the file is 'web: node app.js'. Above the code, it says 'You, 19 hours ago | 1 at'.

## Databases Tier

The database tier is quite simple, since it requires minimal configuration changes to work on a production, live server. Since the application will be on the open internet, the mongodb cluster must also be available from not just the development local ips. In order to fix this, the implementor must allow access from any ip through the network access tab in mongodb atlas. Once on the page, the top right green button, add ip address, will have an option to allow from anywhere. In an ideal scenario, only the deployment ip should be given access, yet heroku assigns dynamic ips that are not trackable, so any ip must be able to access the mongodb cluster. If this change succeeds, the database tier is complete.

mongoDB Atlas All Clusters

CONTEXT: MIGUEL'S ORG - 2020-02-15 > CU-CONNECT

Network Access

IP Whitelist Peering Private Endpoint

+ ADD IP ADDRESS

You will only be able to connect to your cluster from the following list of IP Addresses:

| IP Address  | Comment | Status | Actions     |
|---|---------|--------|-------------|
| 72.226.29.112/32 (includes your current IP address) |         | Active | EDIT DELETE |
| 0.0.0.0/0 (includes your current IP address)        |         | Active | EDIT DELETE |
| 72.94.144.2/32                                      |         | Active | EDIT DELETE |

## Add IP Whitelist Entry

Atlas only allows client connections to a cluster from entries in the project's whitelist. Each entry should either be a single IP address or a CIDR-notated range of addresses. [Learn more.](#)

Whitelist Entry:

Comment:

☐ This entry is temporary and will be deleted in

Cancel Confirm

## Cloud Provider Tier


The final step relates to the cloud provider. Although there are a plethora of options, Heroku offers great performance at no cost. Once the implementer creates an account, he must create a new application. This will lead to a screen that details out deployment steps which are easy to follow. The developer must run these commands inside the commands prompt of the application once git is installed on the machine. If the commands are executed in the proper order, the application will be live and working. The application will be accessible at the following link: **<https://app-name.herokuapp.com/>**


### Create New App

**App name**

app-name

**Choose a region**

 United States

 **Add to pipeline...**

Create app

#### Deploy using Heroku Git

Use git in the command line or a GUI tool to deploy this app.

#### Install the Heroku CLI

Download and install the [Heroku CLI](#).

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

#### Clone the repository

Use Git to clone cu-connect's source code to your local machine.

```
$ heroku git:clone -a cu-connect
$ cd cu-connect
```

#### Deploy your changes

Make some changes to the code you just cloned and deploy them to Heroku using Git.

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```

app.js - Main Project - Visual Studio Code

```
77 app.use( /images , express.static(path.join(__dirname, images )));
78
79 // Function Access And Write To Log File
80 const accessLogStream = fs.createWriteStream(
81   path.join(__dirname, "access.log"),
82   { flags: "a" }
83 );
84
85 // Filter Through Requests For Compression, Security, And Logging
86 app.use(helmet());
87 app.use(compression());
88 app.use(morgan("combined", { stream: accessLogStream }));
89
90 // Filter Requests Through bodyParser/Multer/Session
91 app.use(bodyParser.urlencoded({ extended: true }));
92 app.use(bodyParser.json());
93 app.use(
94   multer({ storage: fileStorage, fileFilter: fileFilter }).single("image")
95 );
96 app.use(
97   session({
```

PROBLEMS 16 OUTPUT DEBUG CONSOLE TERMINAL

1: cmd

Microsoft Windows [Version 10.0.18363.752]  
(c) 2019 Microsoft Corporation. All rights reserved.  
  
C:\Users\mtapi\Miguel\University\Cairn University\Y4 - Senior\1 - Spring\Software Engineering\Main Project>heroku login