

COMPANY RATING WEBSITE

INSTALLATION GUIDE

*Report #6*

|  |  |  |
| --- | --- | --- |
| **COMPANY RATING WEBSITE – *KẾ SINH NHAI TEAM*** | | |
| **Group Members** | Trần Quốc Hưng | SE02970 |
| Nguyễn Sơn Lâm | SE03090 |
| Nguyễn Việt Hùng | SE03293 |
| Nguyễn Nhật Quang | SE03122 |
| Trần Đăng | SE03039 |
| **Supervisor** | Bùi Đình Chiến | |
| **Project code** | CRW | |

**- Hanoi, 28/05/2016 -**

# SIGNATURE PAGE

**AUTHORS**: Nguyễn Sơn Lâm 16/08/2016

*Team member*

**REVIEWER**: Trần Quốc Hưng 16/08/2016

*Project Leader*

**APPROVAL**: Bùi Đình Chiến --/--/2016

*Supervisor*

# Record of change

\*A – Added; M – Modified; D – Deleted

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Effective Date | Changed Item | A,M,D\* | Change Description | Reason for Change | Rev. Number |
| 16/08/2016 | Installation Guide | A | First version | Create Installation Guide | 1.0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Table of Contents

[SIGNATURE PAGE 2](#_Toc459288612)

[Record of change 3](#_Toc459288613)

[1. INTRODUCTION 4](#_Toc459288614)

[1.1 Purpose 4](#_Toc459288615)

[1.2 Viết tắt 4](#_Toc459288616)

[1.3 Environment 5](#_Toc459288617)

[2. INSTALLATION GUIDE 5](#_Toc459288618)

[2.1 Deploy website to AWS 5](#_Toc459288619)

[2.1.1 Setup hosting at AWS 5](#_Toc459288620)

[2.1.1.1 Setup new account to use EC2 instance at AWS 5](#_Toc459288621)

[2.1.1.2 Launching EC2 Instance 6](#_Toc459288622)

[2.1.1.3 Setting Security Access 8](#_Toc459288623)

[2.1.2 Deploying website code onto AWS 10](#_Toc459288624)

[2.1.2.1 Connecting to EC2 Instance 10](#_Toc459288625)

[2.1.2.2 Deploying CRW code onto EC2 Instance 12](#_Toc459288626)

[2.1.2.3 Configuring MongoDB 12](#_Toc459288627)

[2.1.3 Running website 16](#_Toc459288628)

[2.2 Setup RoboMongo at local computer 17](#_Toc459288629)

# INTRODUCTION

## Purpose

This document contains guide-lines step by step to deploy CRW to Amazon Web Service (AWS). This user's guide has been developed to help users better understand the requirements that may apply to use this website by providing them simple step by step tutorials. This manual is intended to help make website's functions more accessible and their associated requirements more understandable to users.

## Viết tắt

|  |  |  |
| --- | --- | --- |
| Từ | Ý nghĩa | Chú thích |
| CRW | Company Rating Website | Tên dự án. |
| SRS | Software Requirements Specification |  |
| Người dùng | Khách, thành viên và quản trị viên |  |
| NDC | Người dùng chính | Thành viên và quản trị viên, là những người đóng góp nội dung trong website |
| QTV | Quản trị viên |  |
| URL | Uniform Resource Locator | Đường dẫn của trang web |

## Environment

Following are the softwares required to start the CRW system:

* Operating System: Windows 8, Window 10
* Browsers: Firefox 40, Chrome 50 or higher.
* Database: MongoDB 3.2
* Tool: RoboMongo 0.9, PuTTY, PuTTYgen

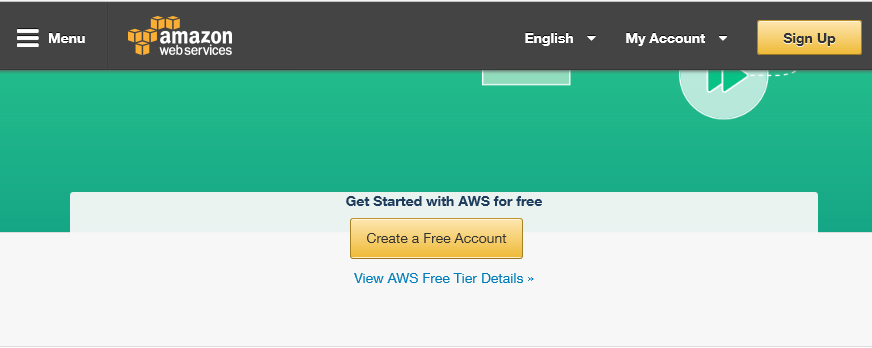
# INSTALLATION GUIDE

## Deploy website to AWS

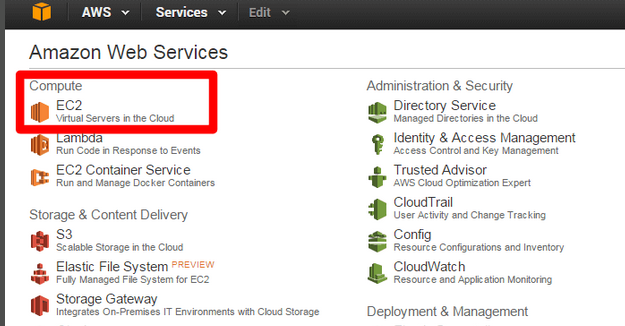
## Setup hosting at AWS

## Setup new account to use EC2 instance at AWS

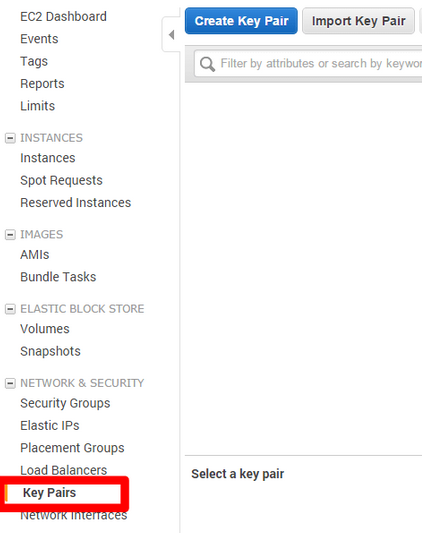
* **Step 1:** Create an account with AWS with link: <https://aws.amazon.com/>



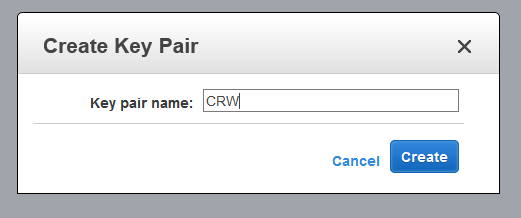
* **Step 2:** Once registered, click “Sign In to the Console”. Click to EC2 at Dashboard



* **Step 3:** Before launching EC2 instance, we need to create an EC2 key pair. Navigate to the menu option titled “Key Pairs” and click “Create Key Pair”.



* **Step 4:** Name key pair to “CRW” in the pop-up that appears. Immediately after, a .pem file with the same name will download to system. Don’t lose this file, it’s very important! It’s only generated once and we’ll need it to access EC2 instance.

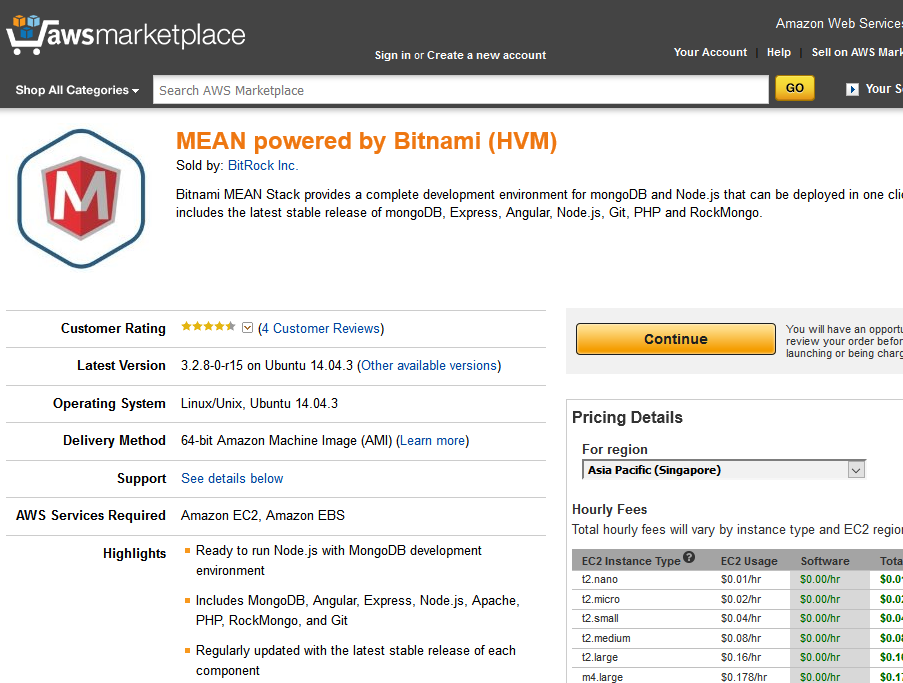


## Launching EC2 Instance

If we lauch EC2 instance using the traditional way, we have to install and config every core components of MEAN, which is very complicated. The better choice is AWS Market place. The Marketplace includes a variety of pre-configured images, include MEAN-supported image, which help us simplifying progress of deploying CRW onto AWS.

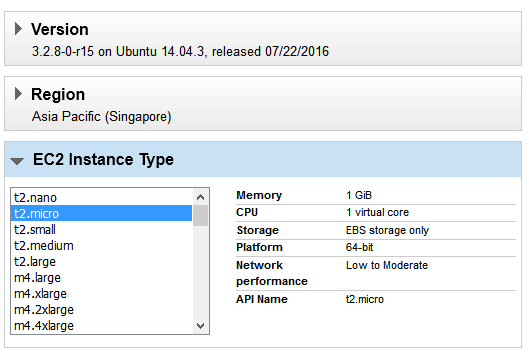
* **Step 1:** Go to AWS Marketplace: <https://aws.amazon.com/marketplace>

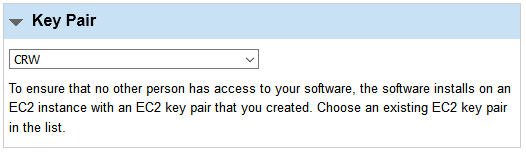
Search for this in the marketplace and open up the one titled: “MEAN Powered by Bitnami (HVM)”.



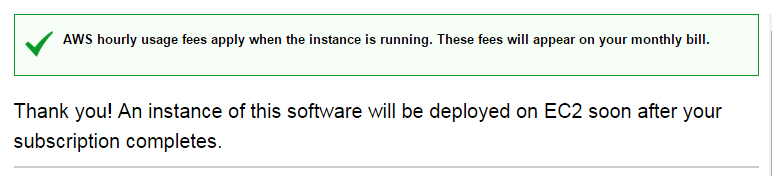
* **Step 2:** From the Bitnami MEAN screen, set region to “Asia Pacific (Singapore)”, click “Continue”
* **Step 3:** Choose suitable instance type based on project budget (t2.micro for free).

At Key Pair section, select “CRW” (the Key Pair we created earlier).





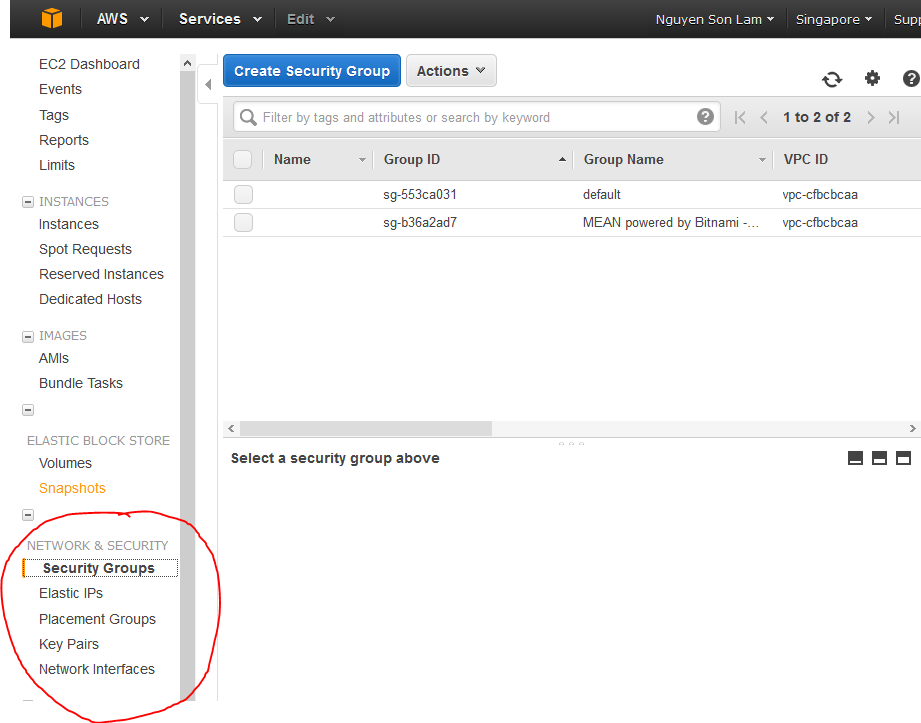
* Step 4: Click “Accept Terms & Launch with 1-Click”. We will then receive a notification telling you that your EC2 instance is being deployed.



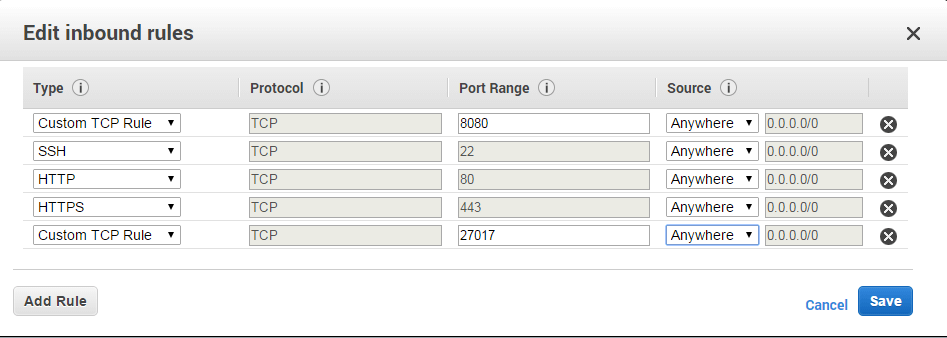
## Setting Security Access

Different from testing local website, whenever we build MEAN applications we designate a specific port to which the server refers. (localhost:3000, localhost:8080, etc). This doesn’t require configuring when we’re running locally, but now that we’re in the real-world, our ports aren’t just open by default. This means since our application is looking to use port 8080, 27017 (for MongoDB) we need to make sure that port 8080 and 27017 accessible for use.

* **Step 1:** Navigate to the AWS EC2 Dashboard. Using the menu on the left, open up the Security Groups panel. Click on the row associated with our MEAN application. (It will have a Group name like: “MEAN powered by Bitnami”). Beneath our row is a set of Inbound and Outbound rules specifying which ports are accessible for the application and by the outside world.



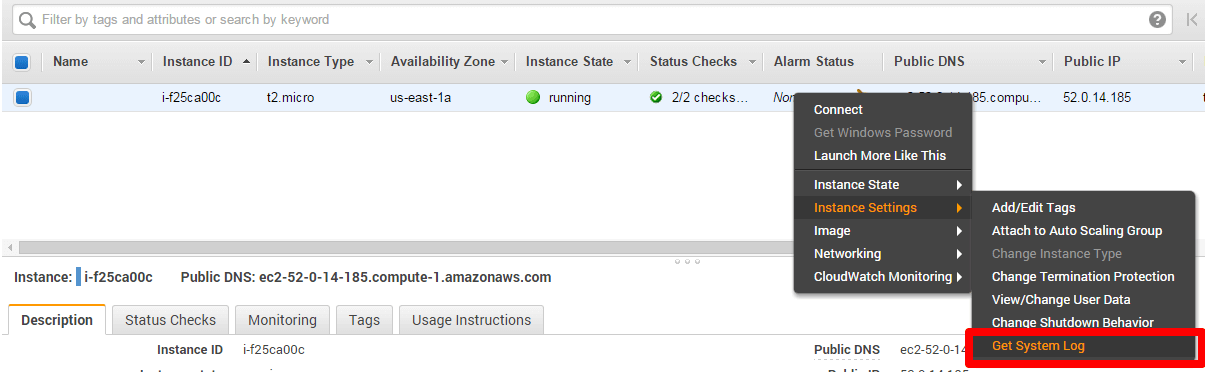
* **Step 2:** Create a custom inbound TCP rule that makes port 8080 accessible. Set the Protocol to TCP, the Port Range to 8080, and the source 0.0.0.0/0 (anywhere). Additionally, we’ll do the same for port 27017, which will be used by MongoDB. Once we’ve added this new rule, refresh the security group using the menu button at the top right.



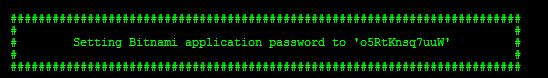
## Deploying website code onto AWS

## Connecting to EC2 Instance

* **Step 1:** *This step is very important*. As part of the Bitnami MEAN installation, EC2 instance comes pre-configured with an initial MongoDB admin username and password. The initial username is always root, but a unique password is created upon first launch of an instance. To obtain password, right-click on the row of EC2 instance and navigate to Instance Settings -> Get System Log.



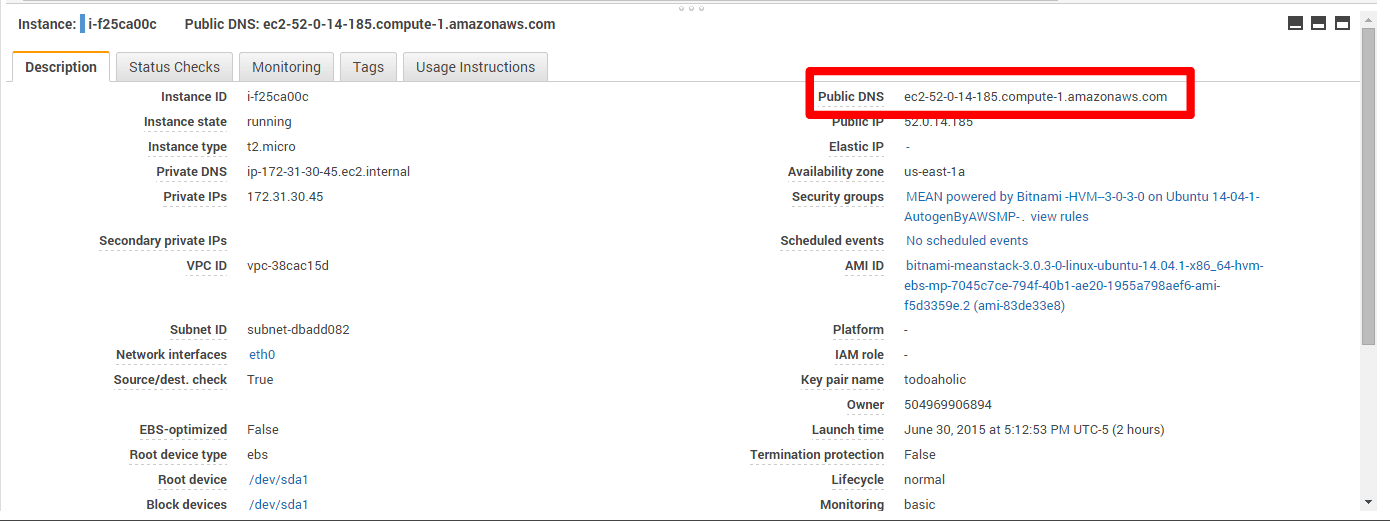
Scroll through the System Log to the very end, we will see a bright box with the words: Setting Bitnami application password to followed by a password. This is the initial MongoDB admin password. Copy it and paste it somewhere safe. *Don’t lose this password*. We will only be shown it once, and we will need this in order to access EC2 instance of MongoDB.



**From step 2 to 4, we will use PuTTY to connect to EC2 instance via SSH.**

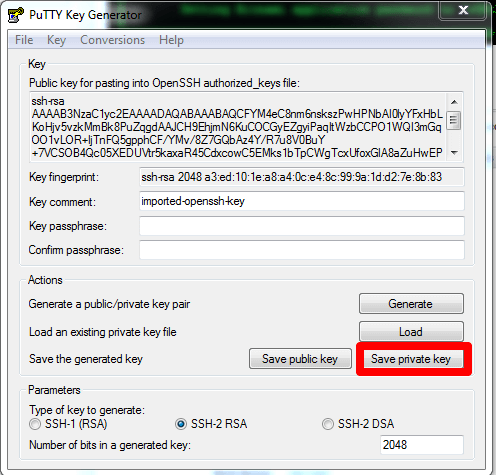
* **Step 2:** Download PuTTY at <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

Once installed, navigate back to AWS Console and to running EC2 instance. From this screen look at the description section associated with EC2 instance. Note the Public DNS. This represents the URL where CRW will sit and the hostname we will be connecting to. we’ll be using this Host Name along with the .pem security key we downloaded earlier to connect to EC2 instance.

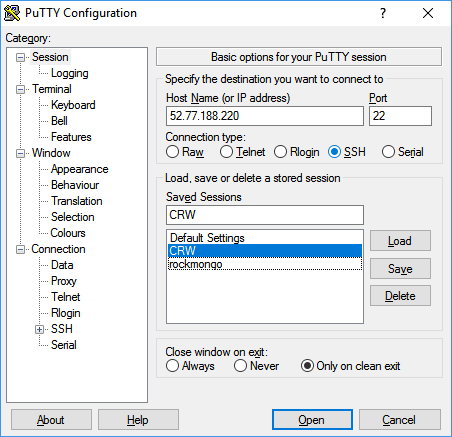
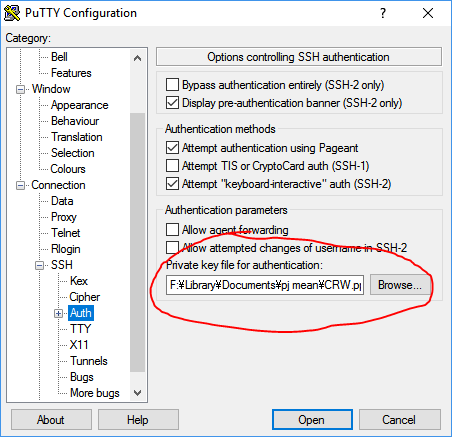


* **Step 3:** Before we can proceed with our connection, we will need to convert .pem file into a .ppk file that PuTTY can recognize. For this we will need the program PuTTYgen (download at <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html> )

Load .pem security file into PuTTYgen (switch the file types to all “All Files .” when browsing for key). Once its loaded, click “Save Private Key” and re-save file as a PuTTY Private Key file (.ppk).



* **Step 4:** Open up PuTTY and enter the Public DNS we noted earlier as the Host Name. Set the Port for 22. Finally, look at the category menu for the SSH -> Auth page. Where it says: “Private Key for Authentication” upload the .ppk file we generated earlier.

Click “Open” and a terminal will pop-up. If this is the first time we’ve accessed this specific host, a warning will appear informing us that the server's host key is not cached in the registry. Just hit “Yes” to instruct it to continue. We’ll then be greeted by a screen saying: “login as”. Enter “bitnami”.



## Deploying CRW code onto EC2 Instance

* **Step 1:** Connect to EC2 Instance via SSH. Type these commands one after another to download CRW code by cloning CRW Git repository onto the server:

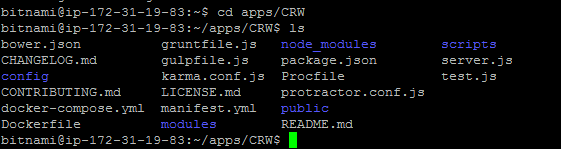
cd

cd apps

sudo git clone <https://github.com/ShiroNek0/CRW.git>

cd CRW

To confirm things worked out, type command *ls* to check that the CRW folder was created. *cd* into that folder and again check the contents with *ls*.



* **Step 2:** Type these commands to install all MEAN.JS dependencies:

sudo npm install -g bower

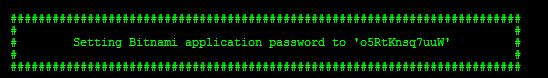
sudo npm install gulp -g

sudo npm install -g grunt-cli

sudo npm install

## Configuring MongoDB

* **Step 1:** Connect to EC2 Instance via PuTTY. When prompted, enter: bitnami to login.
* **Step 2:** We’ll be making use of the green password we obtained earlier:



While in the EC2 terminal, use “mongo admin” to enter MongoDB

sudo mongo admin --username root --password {YOUR GREEN PASSWORD}

Once you’ve logged in, the first action we’ll take is to create a new user with administrative privileges. This will be our admin user.

> show dbs

> use admin

> db.createUser

> db.createUser(

{

user: "AdminCRW",

pwd: "maiwaifu",

roles: [ "root" ]

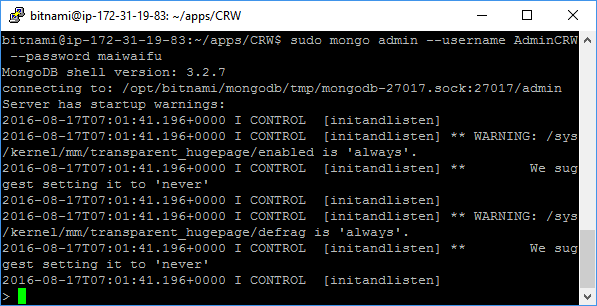
}

)

If all goes well, we should be notified that the user was successfully added.

Now, log out of mongodb by entering ctrl+c and try logging back in using your new admin username and password.

sudo mongo admin --username AdminCRW --password maiwaifu



* **Step 3:** Confirm that your account has been granted complete privileges by querying the databases and adding in a dummy record. Here we’ll add a record: “This is a test” into the “test” collection of “dummyDB”.

> show dbs

> use dummyDB

> db.test.insert({ item: "This is a test"})

If we query for the record we should see it appear.

> db.test.find().pretty()

C:\Users\ShiroNeko\AppData\Local\Microsoft\Windows\INetCacheContent.Word\19.png

**At next step we’ll configure MongoDB so it knows which IP addresses to allow database access.**

* **Step 4:** Head to Stack -> MongoDB by type those commands:

cd

cd stack

ls

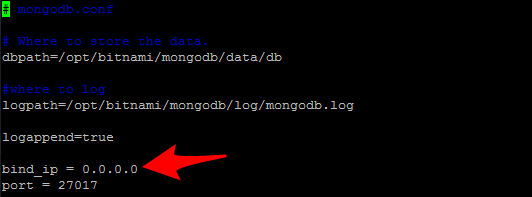
cd mongodb

ls

Open mongodb.conf file and making a few edits using nano editor:

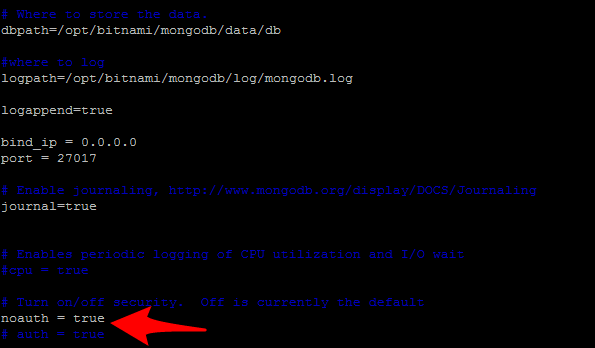
sudo nano mongodb.conf

Change the bind\_ip value to equal 0.0.0.0. This will allow all IP addresses access to our MongoDB instance.



Next, we need to configure the authentication of MongoDB 3.2. With version 3.0, came a new authentication mechanism for MongoDB. In Mongo 2.0, a database could be accessed via a remote shell or terminal so long as an admin username and password was supplied. This was convenient for upstart developers trying to quickly set things up. With the latest change, the authentication mechanism requires a bit more work as many third-party tools (including Bitnami’s default image) have yet to switch to the new drivers.

Look for where it says: auth = true. Temporarily, comment this value out, and uncomment the parameter that says noauth = true. Then press Ctrl+O then Enter to save file, Ctrl+X to exit nano.



Once complete, navigate up one level back to the stack folder. Then run the bash script in ctlscript.sh to restart mongodb.

cd ..

ls

sudo bash ctlscript.sh restart mongodb

After that, login to MongoDB and run the following:

> use admin

> var schema = db.system.version.findOne({"\_id": "authSchema"})

> schema.currentVersion = 3

> db.system.version.save(schema)

Now we create a new admin profile. This administrator will work with the reverted authentication mechanism and work for remote connections.

> show dbs

> use dummyDB

> db.createUser({

user: "AdminCRW\_remote",

pwd: "maiwaifu",

roles:[

{

"role" : "readWrite",

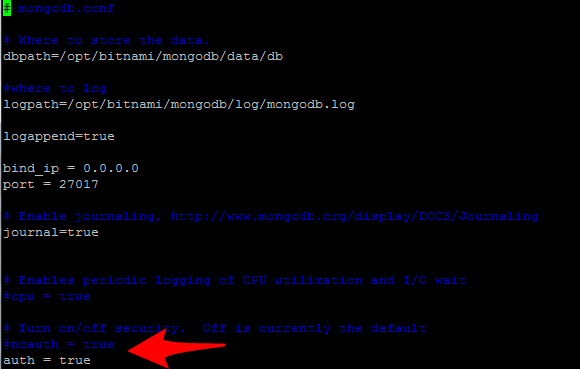
"db": "CRW"

}

]

})

Once created, go back again to the mongodb.conf file, comment out noauth=true, and uncomment auth=true. (Essentially re-activate the authentication requirement). Your config file should now look like this:



Run the bash script in ctlscript.sh to restart mongodb again to ensure changes take effects. We can confirm the changes by connecting to Mongo Instance via local terminal:

mongo {EC2 DOMAIN}/CRW -u {New Username} -p {password}

**At next step, we will incorporate the EC2 DB into our website.**

* **Step 5:** Navigate to the database.js file found in the config folder. Replace the existing URL and replace it with the URL associated with our EC2 instance of MongoDB. It should follow the format below:

mongodb://AdminCRW\_remote:maiwaifu@{EC2 URL}:27017/CRW'

## Running website

* **Step 1:** Connect to EC2 Instance via PuTTY. Navigate to CRW folder

cd

cd apps/CRW

* **Step 2:** As of now, the moment we close out PuTTY, our application dies with it. Conceivably, we could keep our PuTTY window open forever, but that sort of defeats the purpose of having our application on a server.

So instead, we’re going to utilize the node package forever.js. Forever is a simple tool that ensures a given script runs continuously (i.e. forever). To install it, let’s first stop our existing application with ctrl+c.

We’ll use this command:

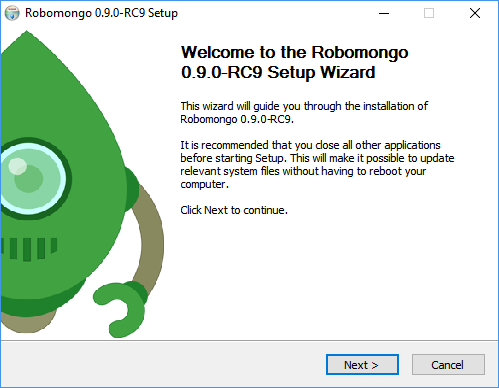
sudo npm install forever -g

We will get a short message saying Forever processing file: server.js and this should be good to go. The best way to test that things after that is to simply close out PuTTY instance of PuTTY and reload website in browser.

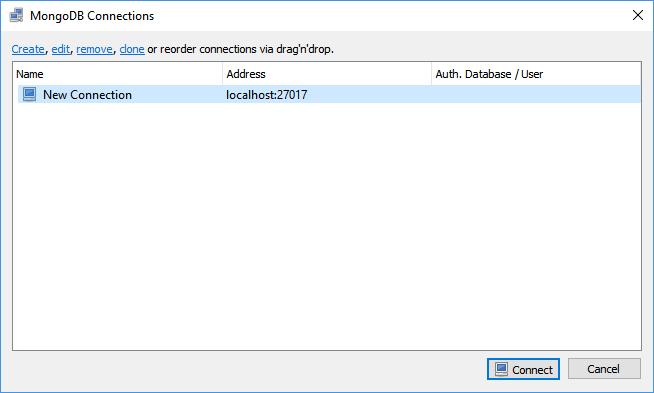
## Setup RoboMongo at local computer

Since Mongo CLI is kind of inconvenient to manage database efficiently, we should use RoboMongo, a native MongoDB management tool, which has a very easy-to-use UI.

* **Step 1:** Download and install RoboMongo at <https://robomongo.org/download>

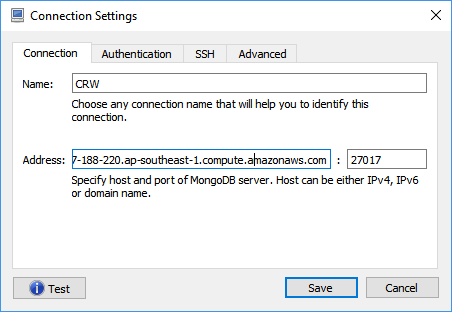


* **Step 2:** Open RoboMongo, press Ctrl+N



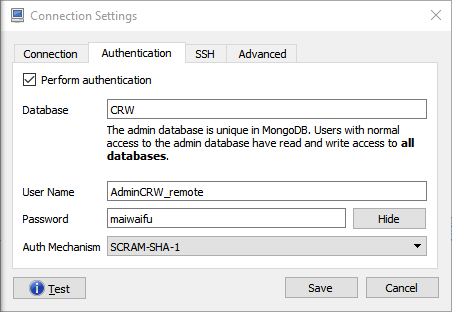
Click “Create” hyperlink to open new Connection Setting.

At tab Connection, set Name to CRW and fill in public domain DNS of EC2 instance to Address textbox.

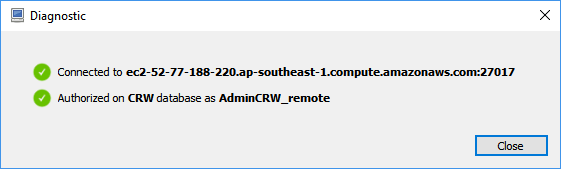


At tab Authentication tab, check “Perform authentication” checkbox, fill in all nescessary info:

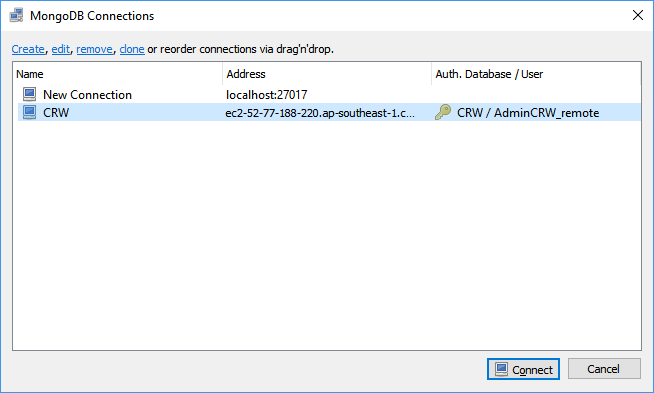
* Database: CRW
* Username: AdminCRW\_remote
* Password: maiwaifu
* Auth Mechanism: SCRAM-SHA-1



At Advanced tab, type CRW. Then click Test to confirm connection to database on EC2 Instance.



* **Step 3:** After confirmation, click Save. Press Ctrl+N, now CRW is displayed on connection list.



Click Connect to access to database.

