

Deploy Node.js app to Amazon EC2

Node.js application can be hosted by various web servers, Apache, IIS, Nginx, etc. In the post, I introduce how to deploy Node.js application to the Amazon Cloud Server, EC2.

There are mainly 5 steps:

- Create EC2 Instance
- Use Putty to Connect EC2 Remote Server
- Setup Node.js environment in EC2 Instance
- Create simple node app and start Node server
- Deploy Local Node.js application to EC2 Instance

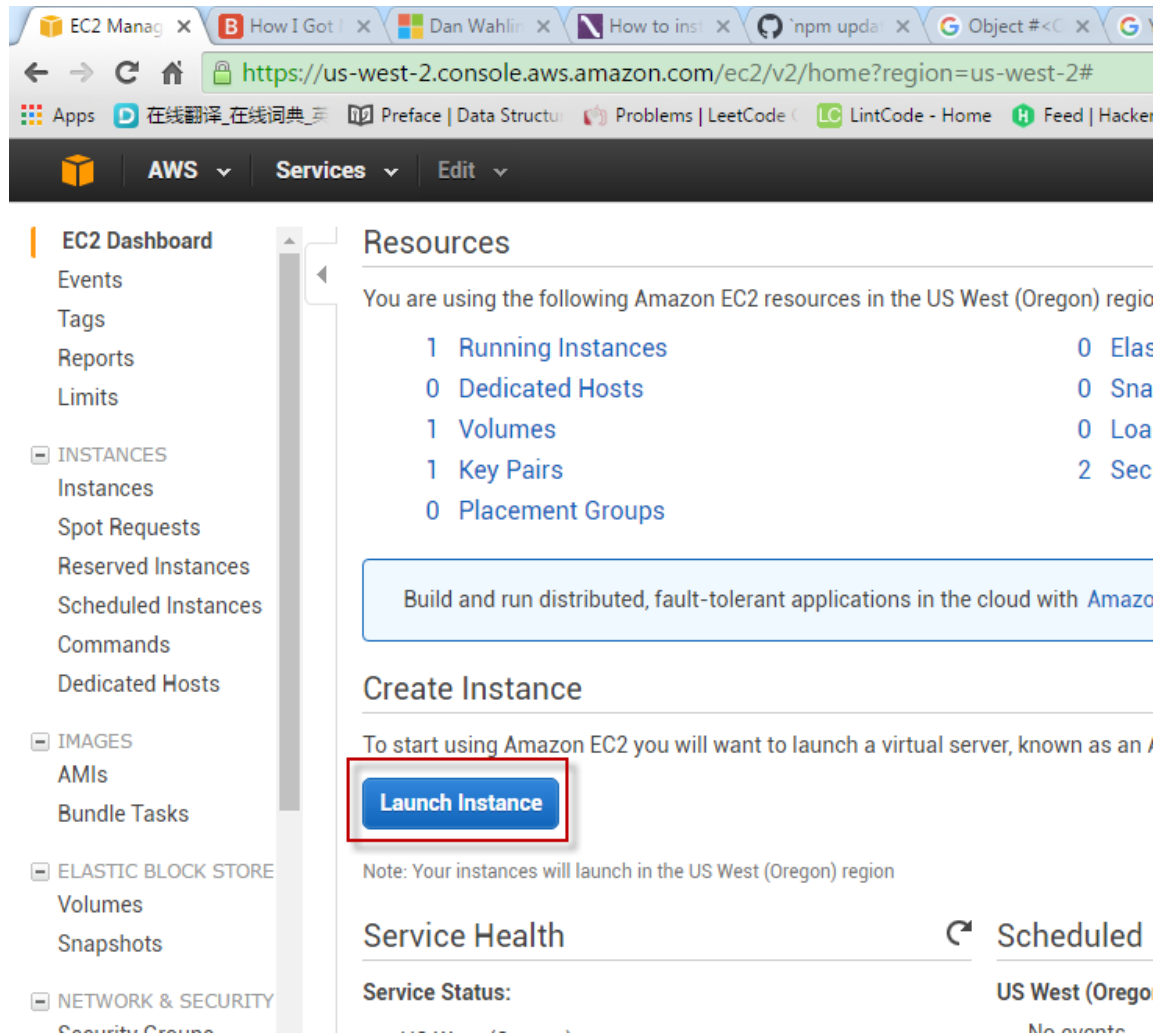
1. Create EC2 Instance

1.1 Login to Amazon EC2

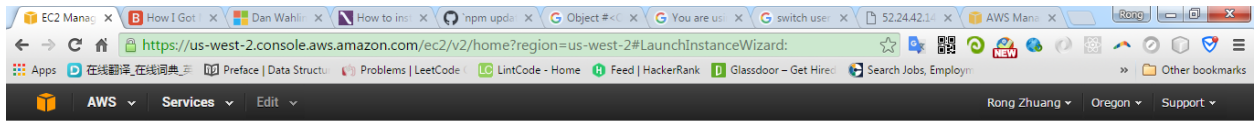
<http://aws.amazon.com/ec2/>

Sign up 'Amazons free micro instance of EC2' if you have no AWS Account yet.

1.2 Create Instance



The screenshot shows the AWS Management Console interface. The browser address bar displays <https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#>. The left sidebar contains the navigation menu with categories like EC2 Dashboard, INSTANCES, IMAGES, ELASTIC BLOCK STORE, and NETWORK & SECURITY. The main content area shows the 'Resources' section for the US West (Oregon) region, listing 1 Running Instance, 0 Dedicated Hosts, 1 Volumes, 1 Key Pairs, and 0 Placement Groups. Below this, there is a 'Create Instance' section with a 'Launch Instance' button highlighted by a red box. A note indicates that instances will launch in the US West (Oregon) region. The 'Service Health' section shows the status for 'Scheduled' events in the 'US West (Oregon) Region'.

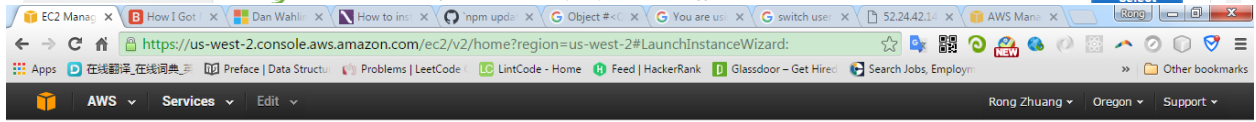


Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start 1 to 22 of 22 AMIs

- Amazon Linux** Free tier eligible
Amazon Linux AMI 2016.03.1 (HVM, SSD Volume Type - ami-d0f506b0)
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs Virtualization type: hvm
Select 64-bit
- Red Hat** Free tier eligible
Red Hat Enterprise Linux 7.2 (HVM, SSD Volume Type - ami-775e4f16)
Red Hat Enterprise Linux version 7.2 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm
Select 64-bit
- SUSE Linux Enterprise Server 12 SP1 (HVM), SSD Volume Type - ami-d2627db3**
Select



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

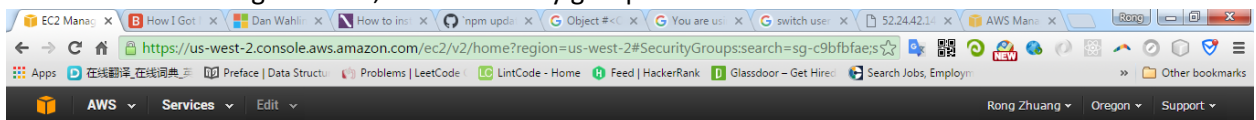
Filter by: All instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
<input checked="" type="checkbox"/>	General purpose	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate

Finally, review and launch.

After the instance is generated, create security group.



Create Security Group **sg-c9bfbfae** **launch-wizard-1**

Name	Group ID	Group Name	VPC ID	Description
sg-c9bfbfae	launch-wizard-1	launch-wizard-1	vp-33d37a57	launch-wizard-1 created 2016-05-05T15:07:45.759-05:00

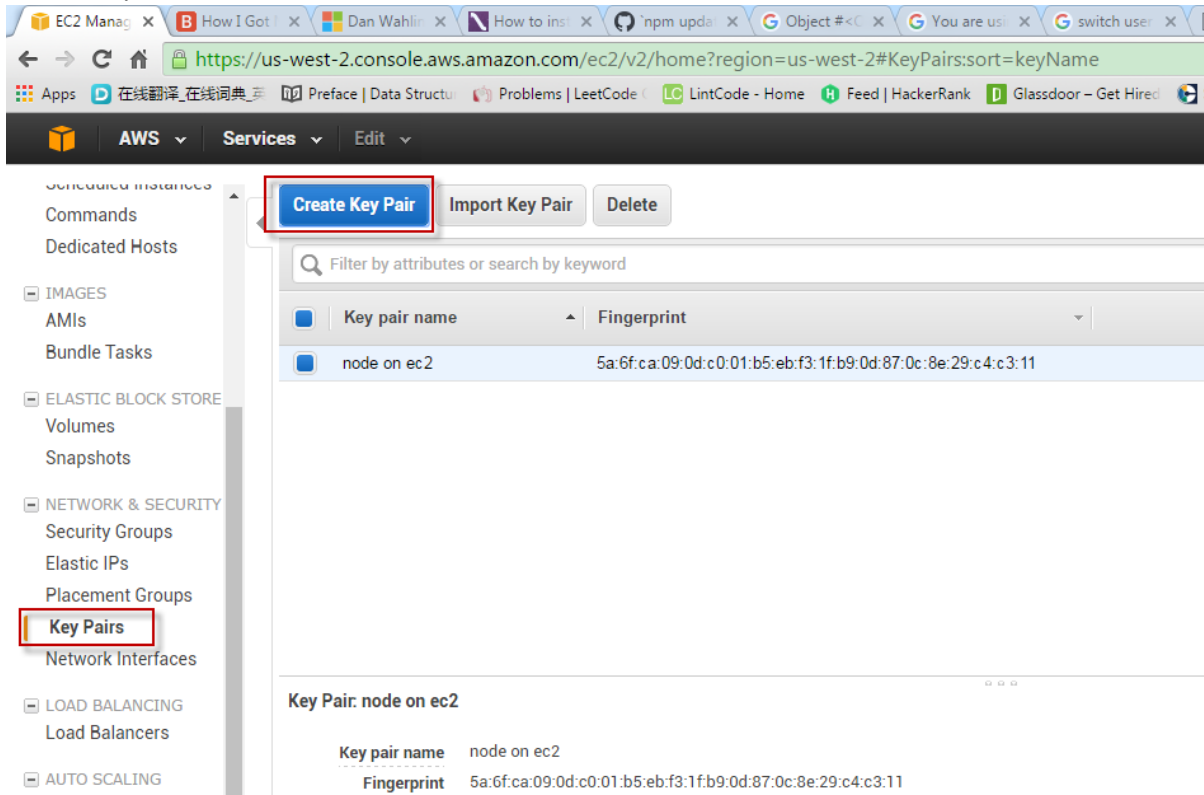
Security Group: sg-c9bfbfae

Description Inbound Outbound Tags

Edit

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0

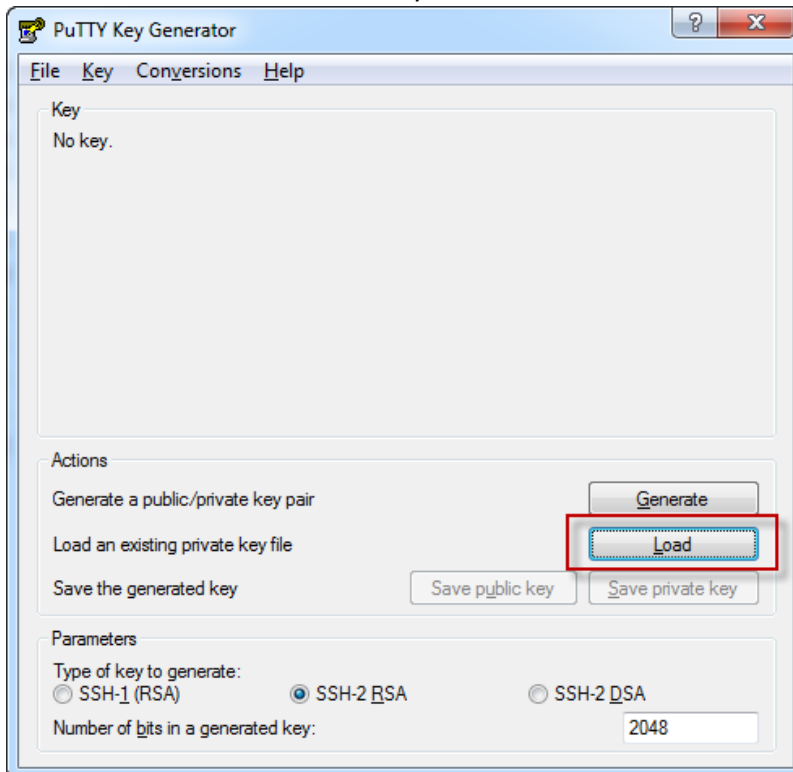
1.3 Create Key Pair

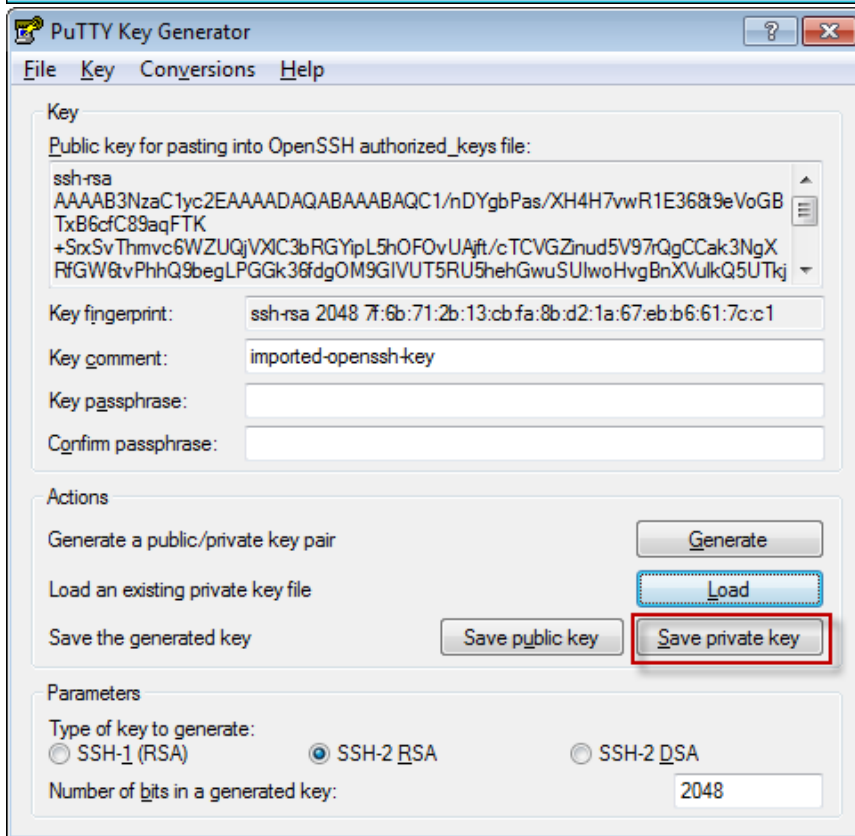
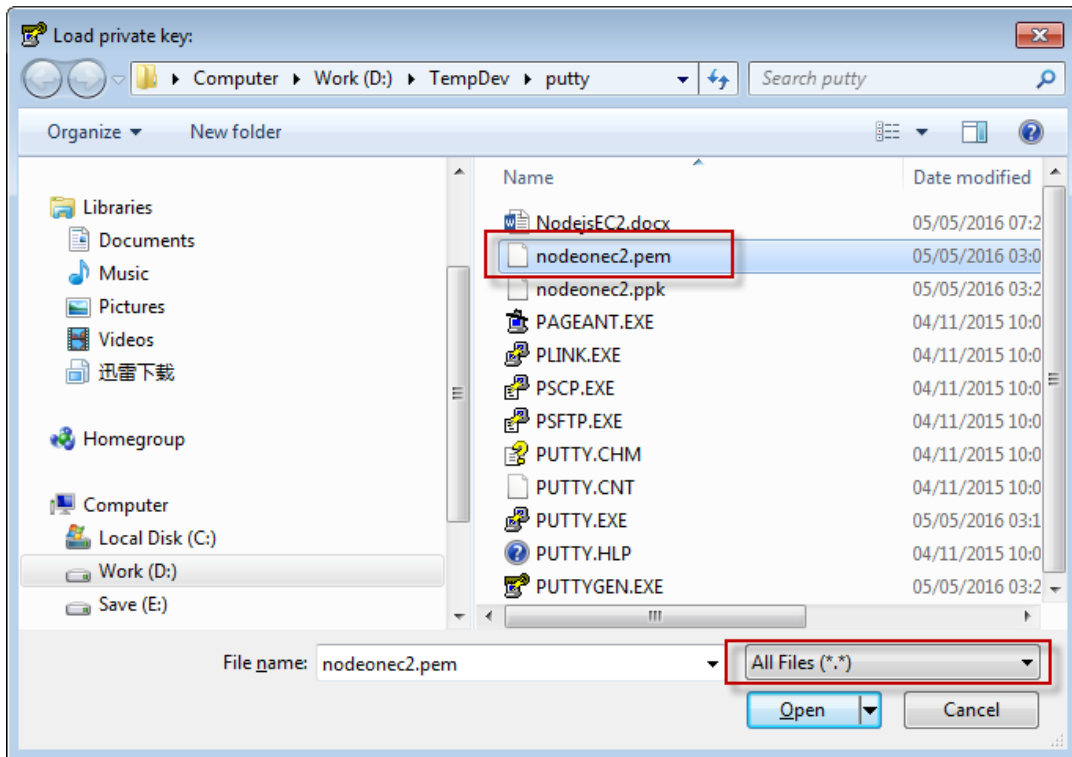


Download the private key to local machine, eg. nodeonec2.pem.

2. Use Putty to Connect EC2 Remote Server

2.1 User PUTTYGEN.EXE to convert key

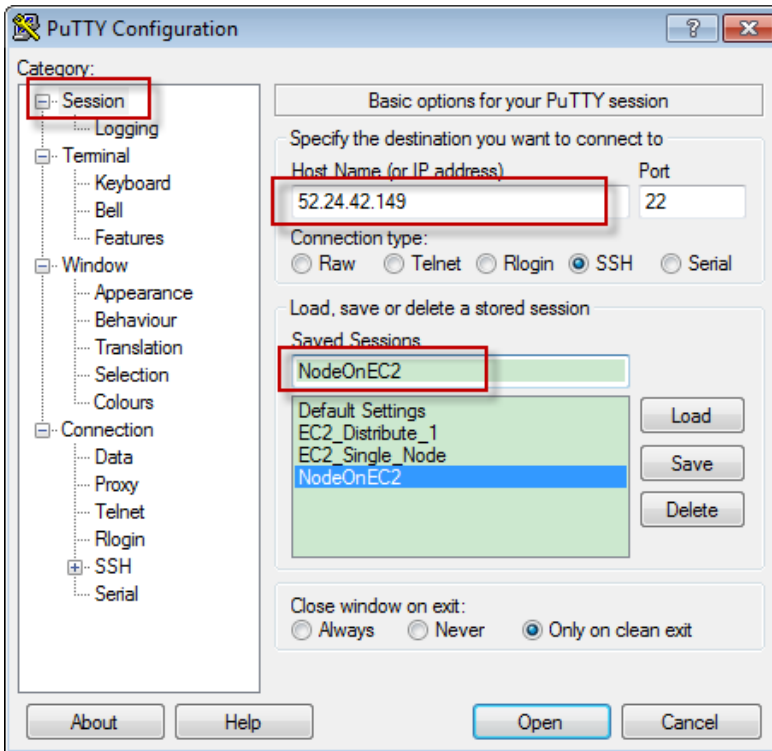




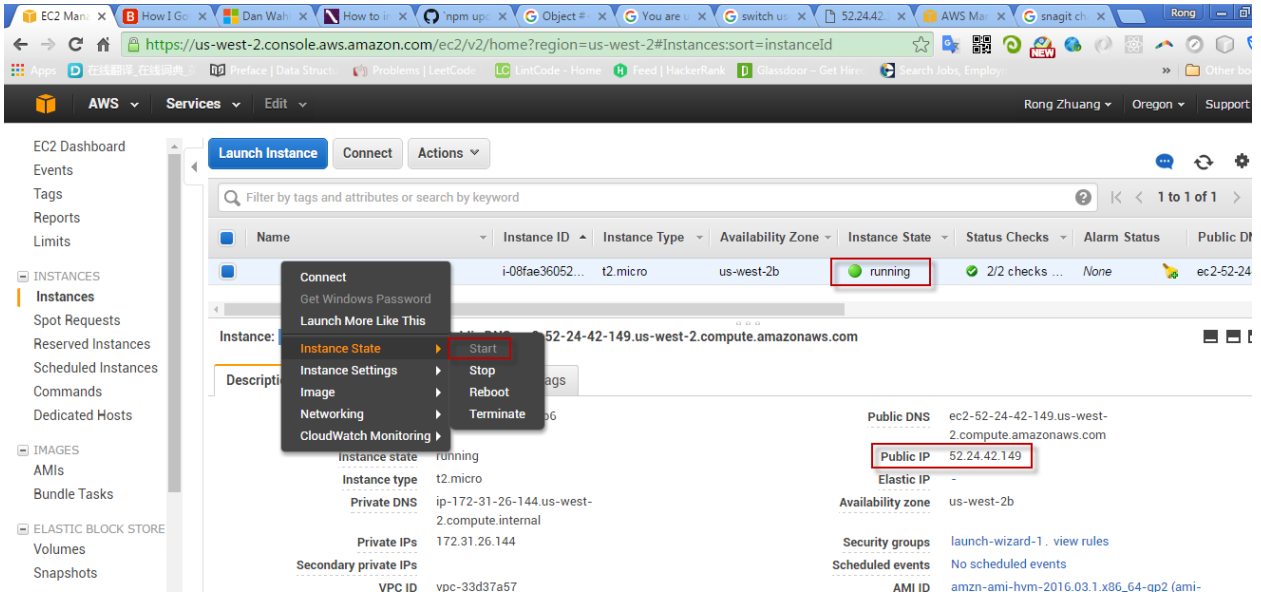
After save, nodeonec2.ppk is generated.

2.2 Configure PuTTY.EXE

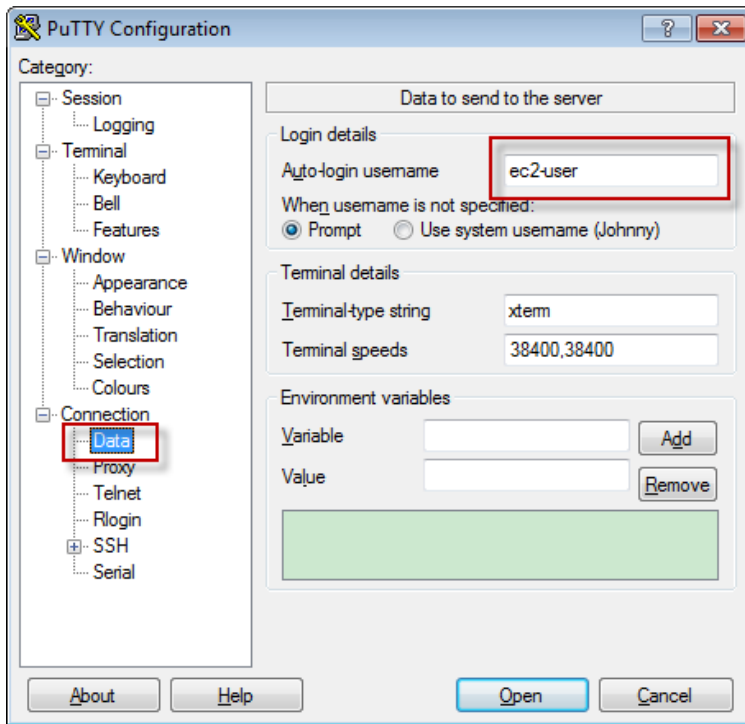
In session, provide IP address(The public ip of your EC2 instance) and session name.



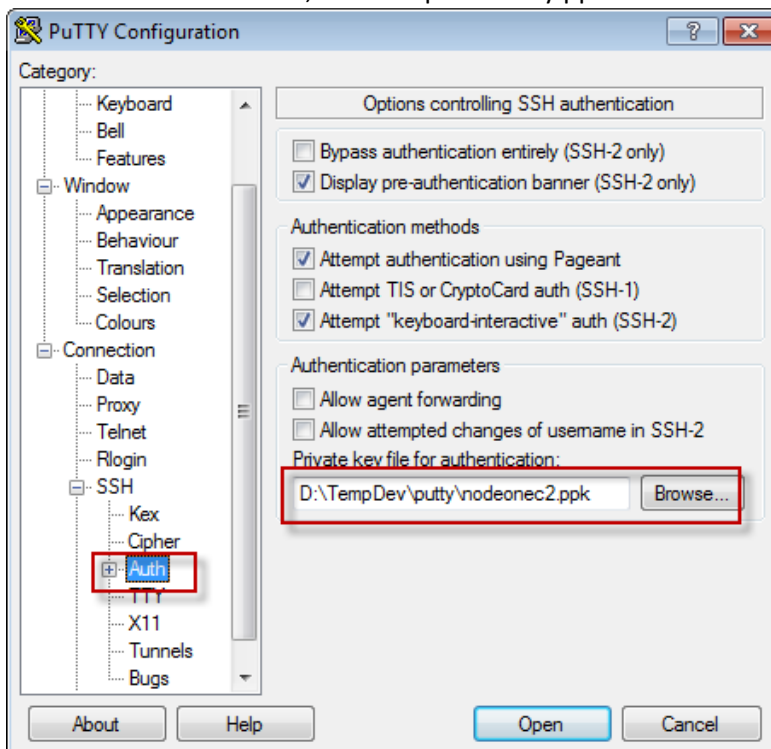
The IP is the Public IP of your EC2 instance. It is only available when the instance is running.



Connection->Data, add user, always 'ec2-user'.



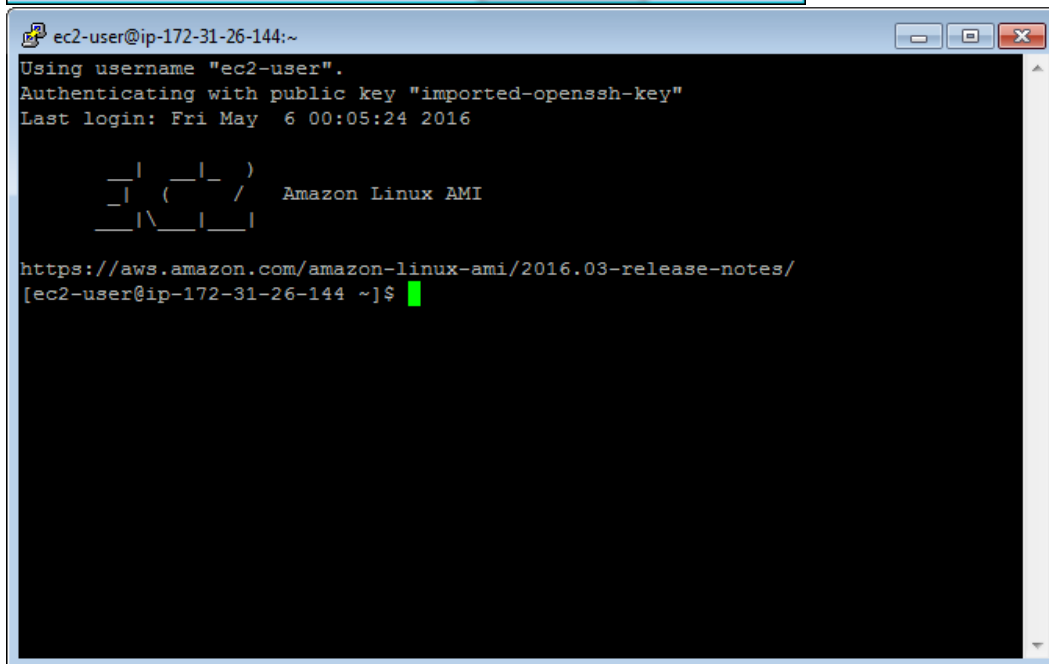
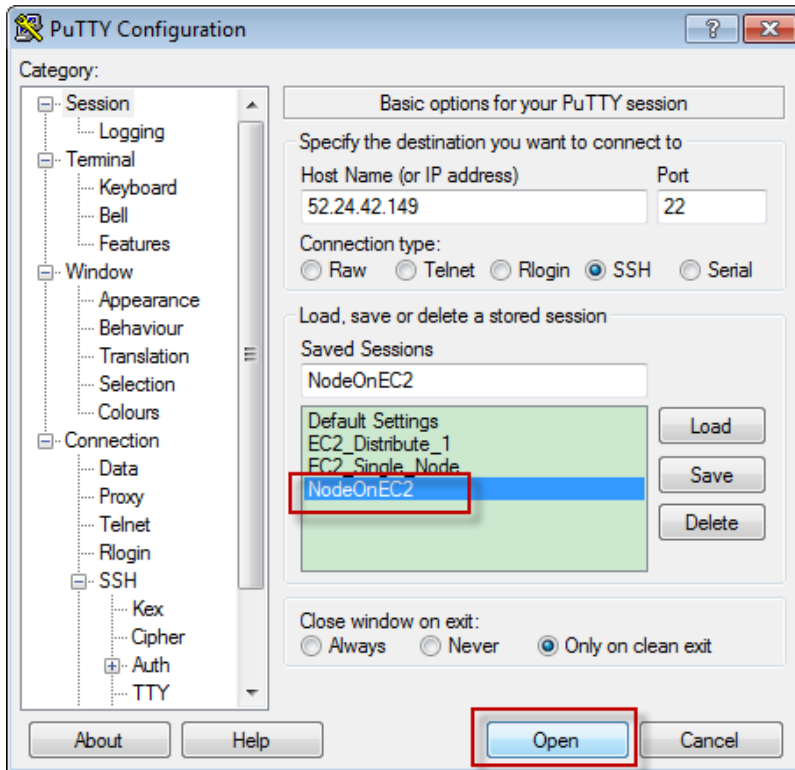
Connection ->SSH ->Auth, load the private key ppk file.



Back to session, save the configuration.

2.3 Connect to EC2 remote server

Choose the newly created session, double click it or click the 'Open' button.



Note, the IP address here is internal IP. When using putty to connect EC2 remote server, make sure launch the instance first. You have to change the IP in putty every time if you reboot the instance. The Public IP address of the EC2 instance changes to different value once it restarts.

3. Setup Node.js environment in EC2 Instance
 - 3.1 Update your EC2 Amazon Linux
sudo yum update
 - 3.2 Install GCC

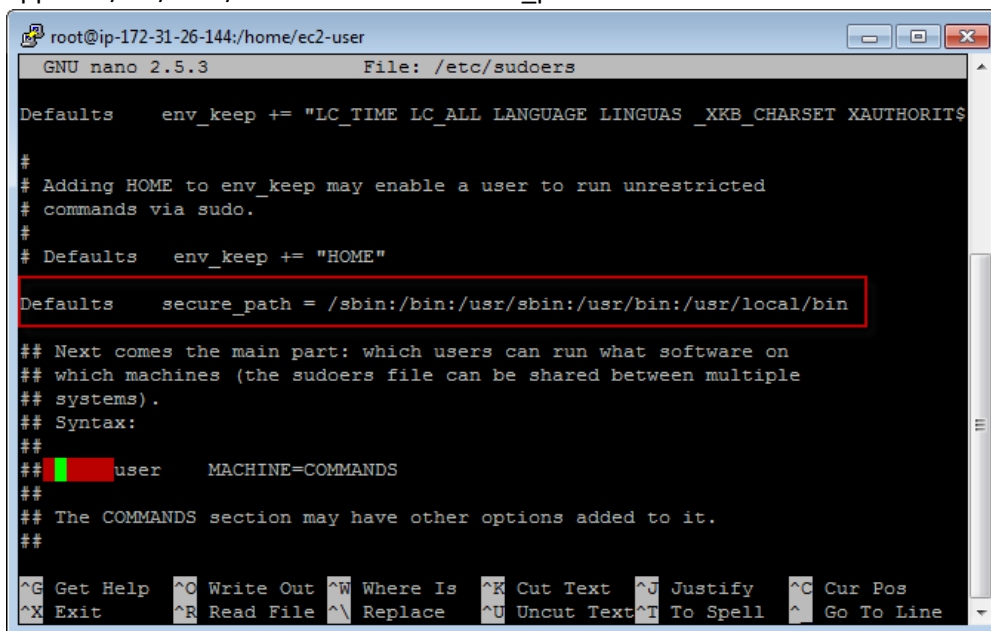
```
sudo yum install gcc-c++ make
sudo yum install openssl-devel
```

3.3 Install Node.js

```
sudo yum install git
git clone git://github.com/nodejs/node
cd node
./configure
make //it may take long time to compile
sudo make install
```

3.4 Add node folder to secure_path

```
sudo su
nano /etc/sudoers
append :/usr/local/bin to the end of secure_path
```



```
root@ip-172-31-26-144:/home/ec2-user
GNU nano 2.5.3 File: /etc/sudoers
Defaults env_keep += "LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY$
#
# Adding HOME to env_keep may enable a user to run unrestricted
# commands via sudo.
#
# Defaults env_keep += "HOME"
Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin
## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
## user MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

3.5 Install npm

```
git clone https://github.com/npm/npm
cd npm
sudo make install
```

4. Create simple node app and start Node server

4.1 Create folder 'site'

```
mkdir site
```

4.2 Create file 'server.js'

```
nano server.js
append the following content to the file, save and exit.
```

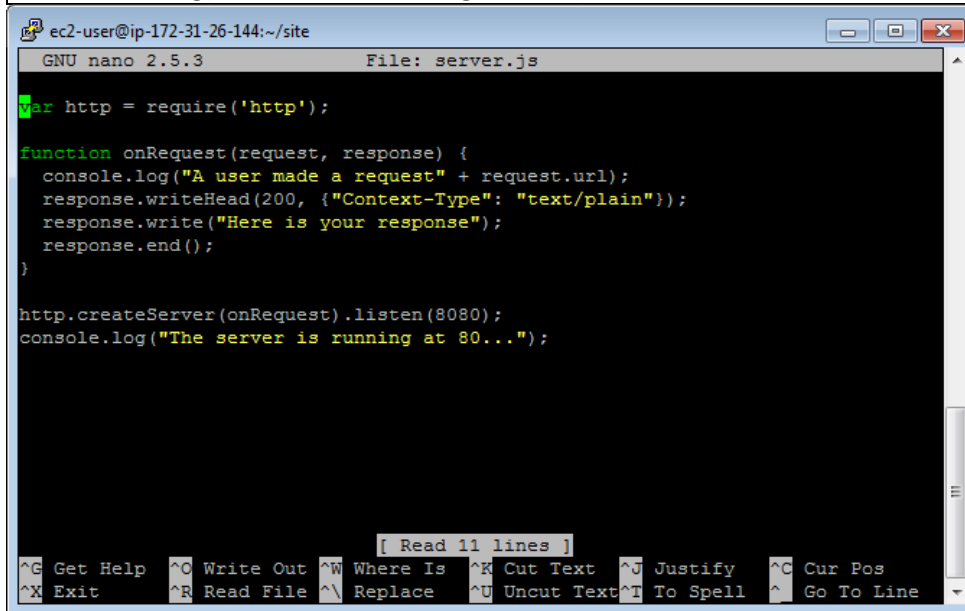
```
var http = require('http');

function onRequest(request, response) {
  console.log("A user made a request" + request.url);
  response.writeHead(200, {"Context-Type": "text/plain"});
```



```
response.write("Here is your response");
response.end();
}

http.createServer(onRequest).listen(8080);
console.log("The server is running at 80...");
```



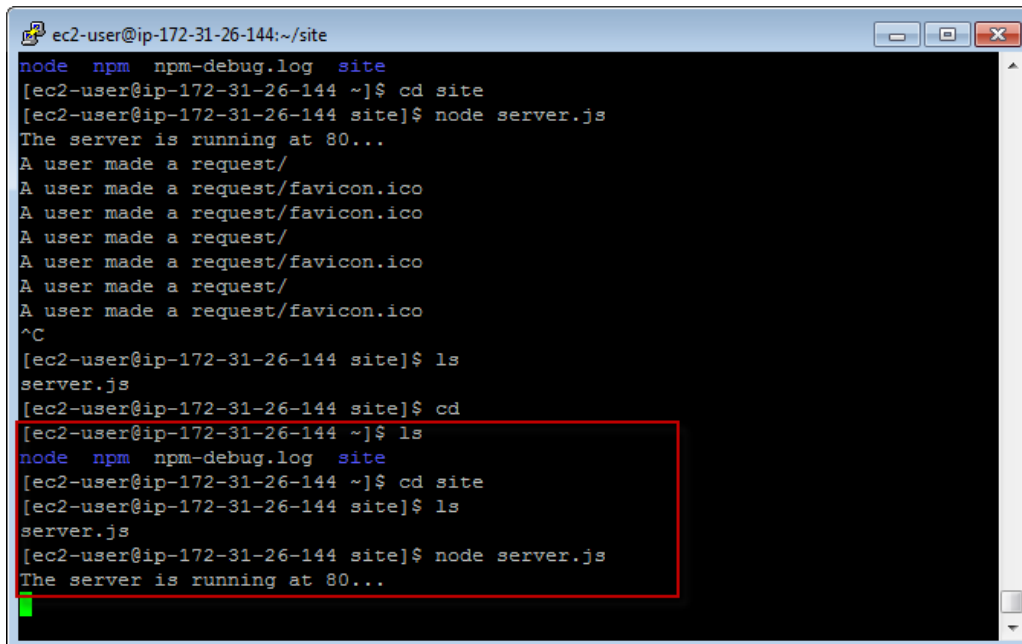
The screenshot shows a terminal window with the nano editor open to a file named 'server.js'. The code in the editor is identical to the code block above. The terminal title bar shows 'ec2-user@ip-172-31-26-144:~/site'. The nano editor interface includes a status bar at the bottom with various keyboard shortcuts like '^G Get Help', '^O Write Out', etc.

4.3 Redirect port

You cannot make node server listen to port 80. Run the following command to redirect requests from port 80 of EC2 server to port 8080 of our Node server.

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to 8080
```

4.4 Start our Node server

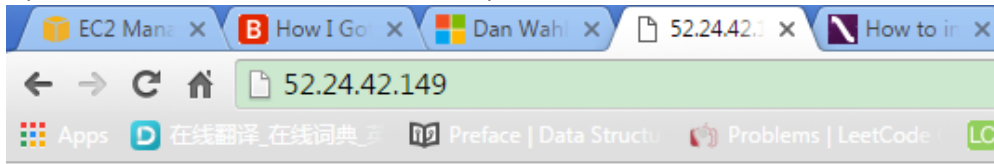


The screenshot shows a terminal window with the following sequence of commands and output:

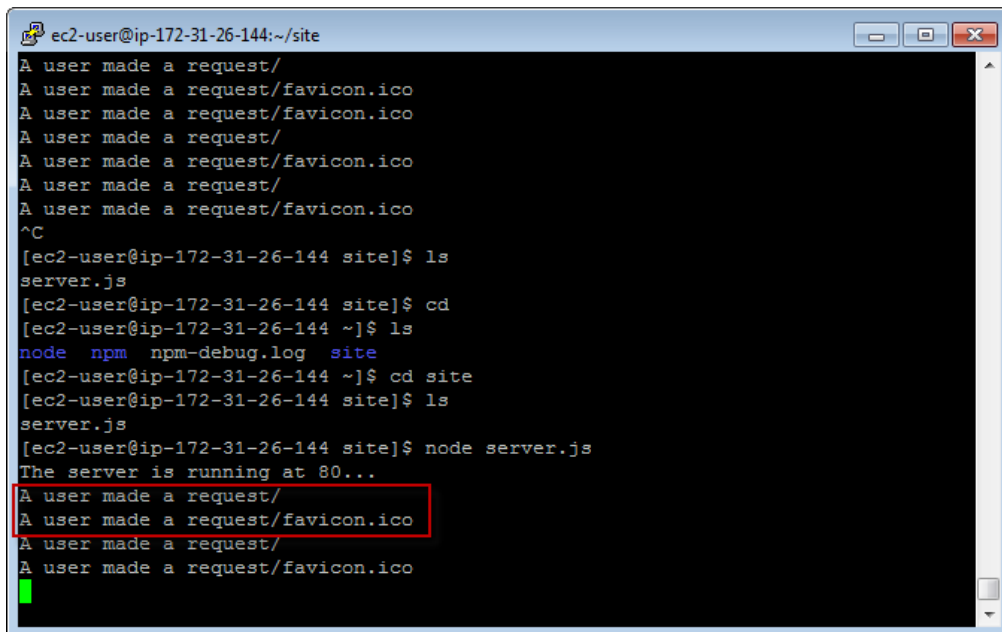
```
node npm npm-debug.log site
[ec2-user@ip-172-31-26-144 ~]$ cd site
[ec2-user@ip-172-31-26-144 site]$ node server.js
The server is running at 80...
A user made a request/
A user made a request/favicon.ico
A user made a request/favicon.ico
A user made a request/
A user made a request/favicon.ico
A user made a request/
A user made a request/favicon.ico
^C
[ec2-user@ip-172-31-26-144 site]$ ls
server.js
[ec2-user@ip-172-31-26-144 site]$ cd
[ec2-user@ip-172-31-26-144 ~]$ ls
node npm npm-debug.log site
[ec2-user@ip-172-31-26-144 ~]$ cd site
[ec2-user@ip-172-31-26-144 site]$ ls
server.js
[ec2-user@ip-172-31-26-144 site]$ node server.js
The server is running at 80...
```

A red box highlights the final sequence of commands and output, showing the server being started again after navigating back to the home directory.

4.5 Open web browser, access the site with public IP.



Here is your response

A screenshot of a terminal window. The prompt is 'ec2-user@ip-172-31-26-144:~/site'. The terminal shows the following commands and output:

```
ec2-user@ip-172-31-26-144:~/site$ ls
server.js
ec2-user@ip-172-31-26-144:~/site$ cd
ec2-user@ip-172-31-26-144:~$ ls
node npm npm-debug.log site
ec2-user@ip-172-31-26-144:~$ cd site
ec2-user@ip-172-31-26-144:~/site$ ls
server.js
ec2-user@ip-172-31-26-144:~/site$ node server.js
The server is running at 80...
A user made a request/
A user made a request/favicon.ico
A user made a request/
A user made a request/favicon.ico
A user made a request/
A user made a request/favicon.ico
```

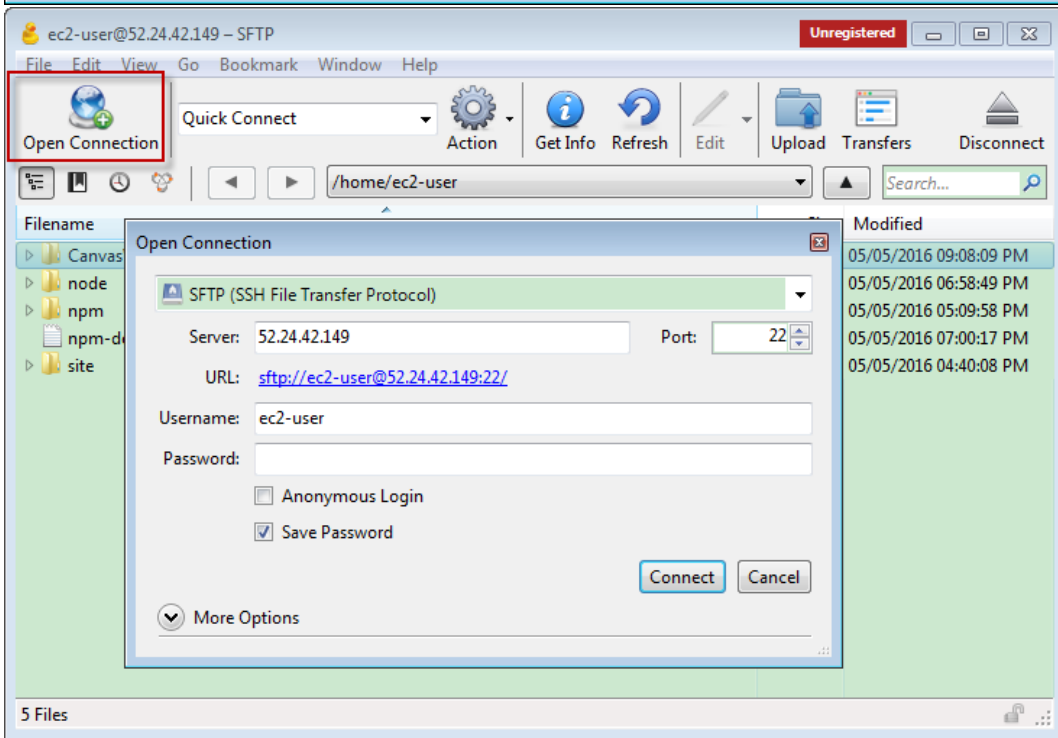
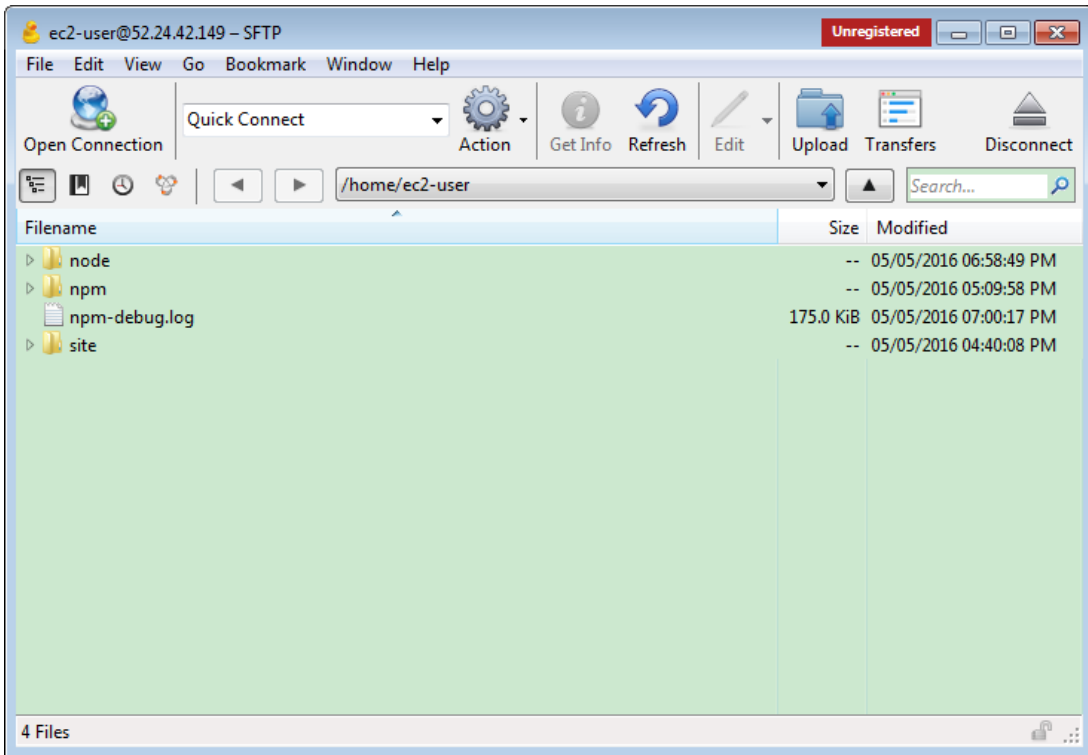
The last four lines of output are highlighted with a red box.

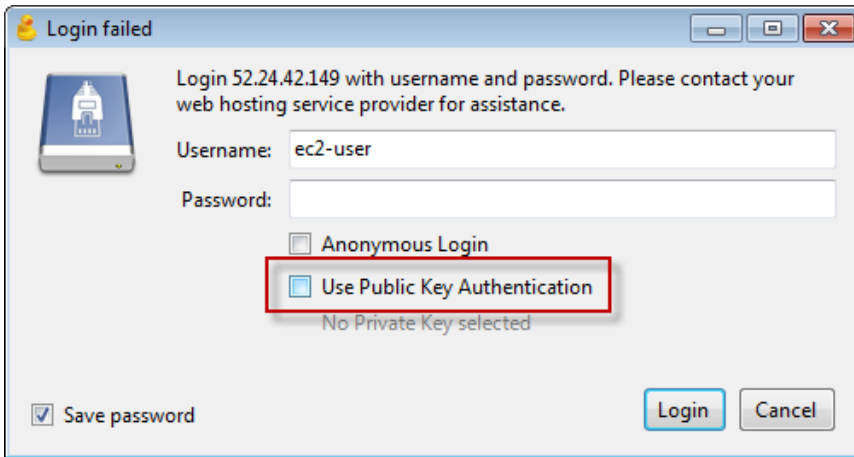
5. Deploy Local Node.js application to EC2 Instance

5.1 Install CyberDuck

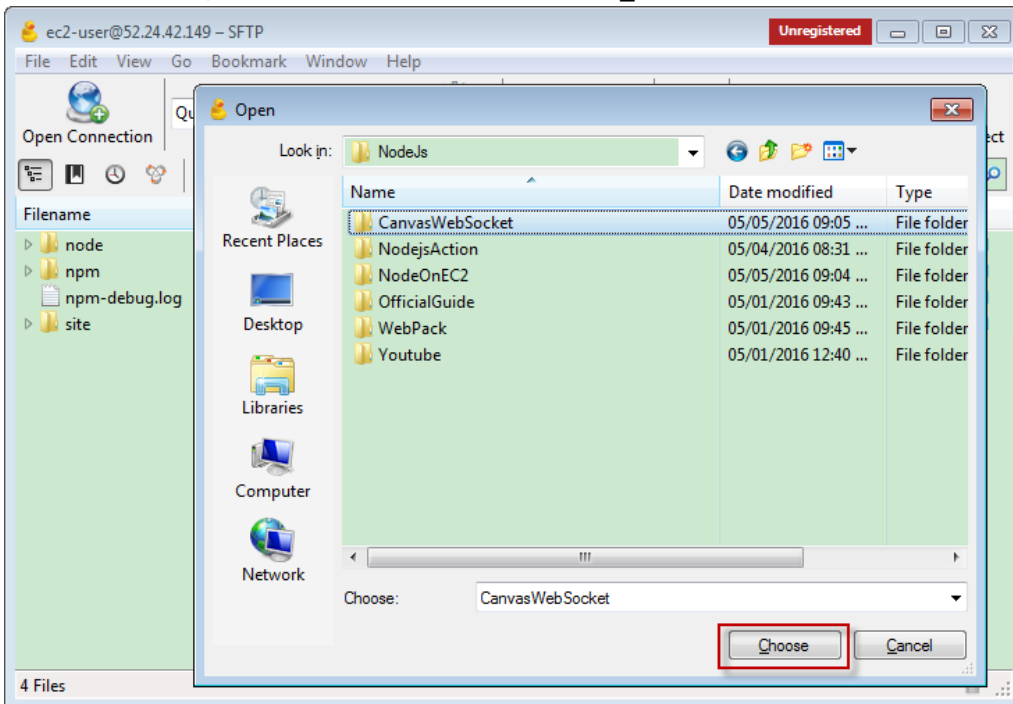
<https://cyberduck.io/?l=en>

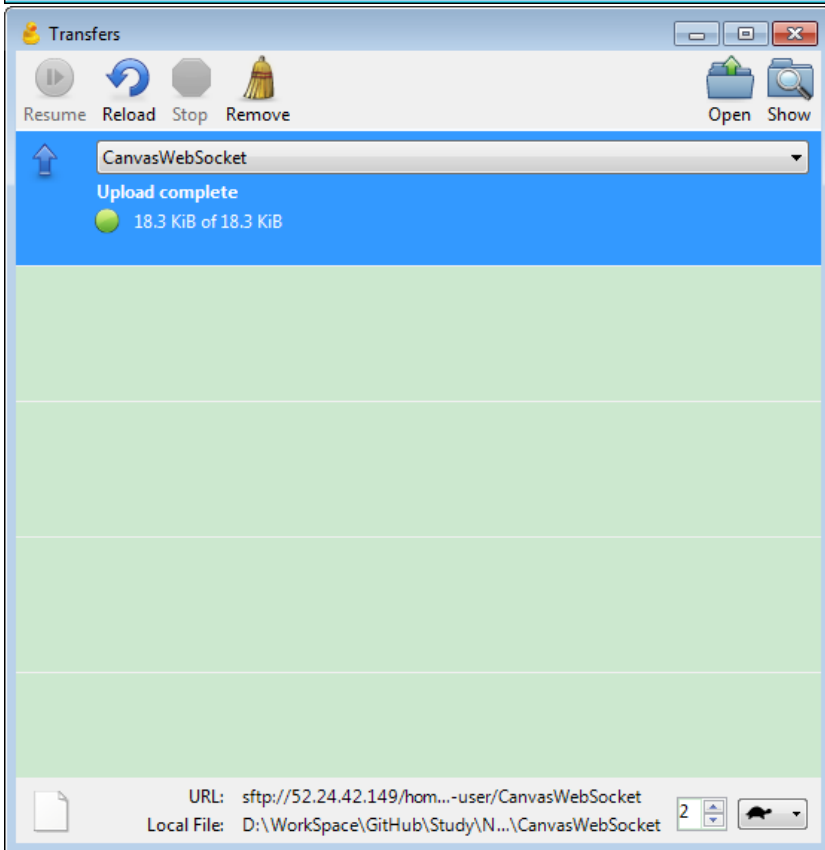
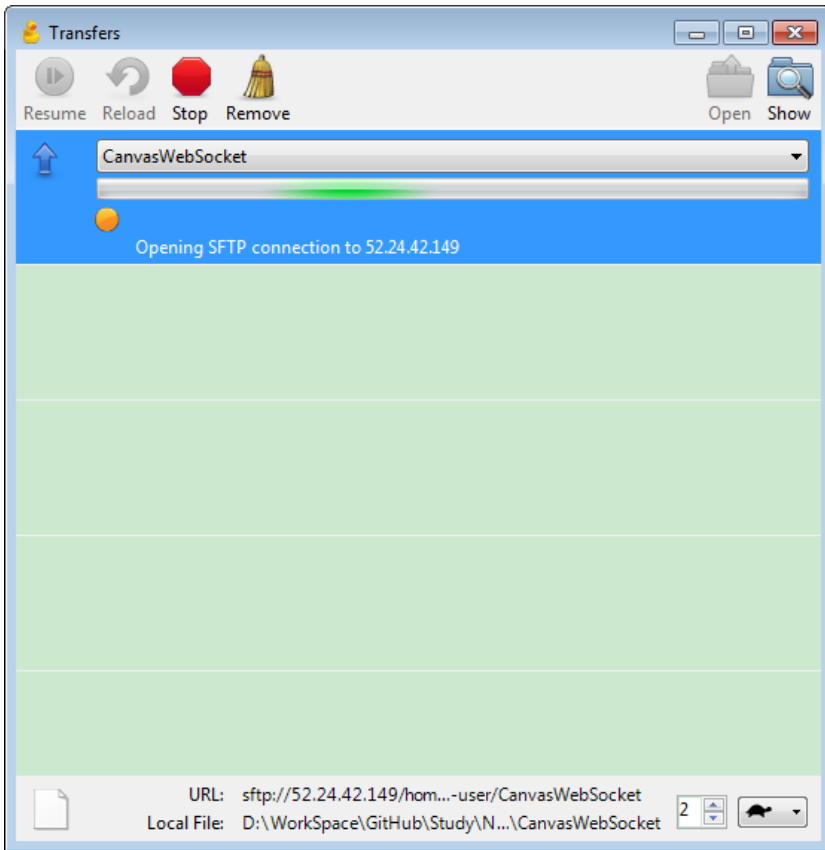
5.2 Launch CyberDuck and Upload





Select the folder, make sure delete all files in 'node_modules' folder.





Refresh the folder in putty, the new folder exists.

```
ec2-user@ip-172-31-26-144:~
--session-command <command>    pass a single command to the shell with -c
                                and do not create a new session
-f, --fast                      pass -f to the shell (for csh or tcsh)
-s, --shell <shell>           run shell if /etc/shells allows it

-h, --help    display this help and exit
-V, --version output version information and exit

For more details see su(1).
[root@ip-172-31-26-144 ec2-user]# sudo su -- ec2-user
[ec2-user@ip-172-31-26-144 ~]$ ls
node  npm  npm-debug.log  site
[ec2-user@ip-172-31-26-144 ~]$ ls
node  npm  npm-debug.log  site
[ec2-user@ip-172-31-26-144 ~]$ cd site
[ec2-user@ip-172-31-26-144 site]$ ls
server.js
[ec2-user@ip-172-31-26-144 site]$ nano server.js
[ec2-user@ip-172-31-26-144 site]$ ls
server.js
[ec2-user@ip-172-31-26-144 site]$ cd
[ec2-user@ip-172-31-26-144 ~]$ ls
CanvasWebSocket  node  npm  npm-debug.log  site
[ec2-user@ip-172-31-26-144 ~]$
```

5.3 Go into the folder

npm install

npm start

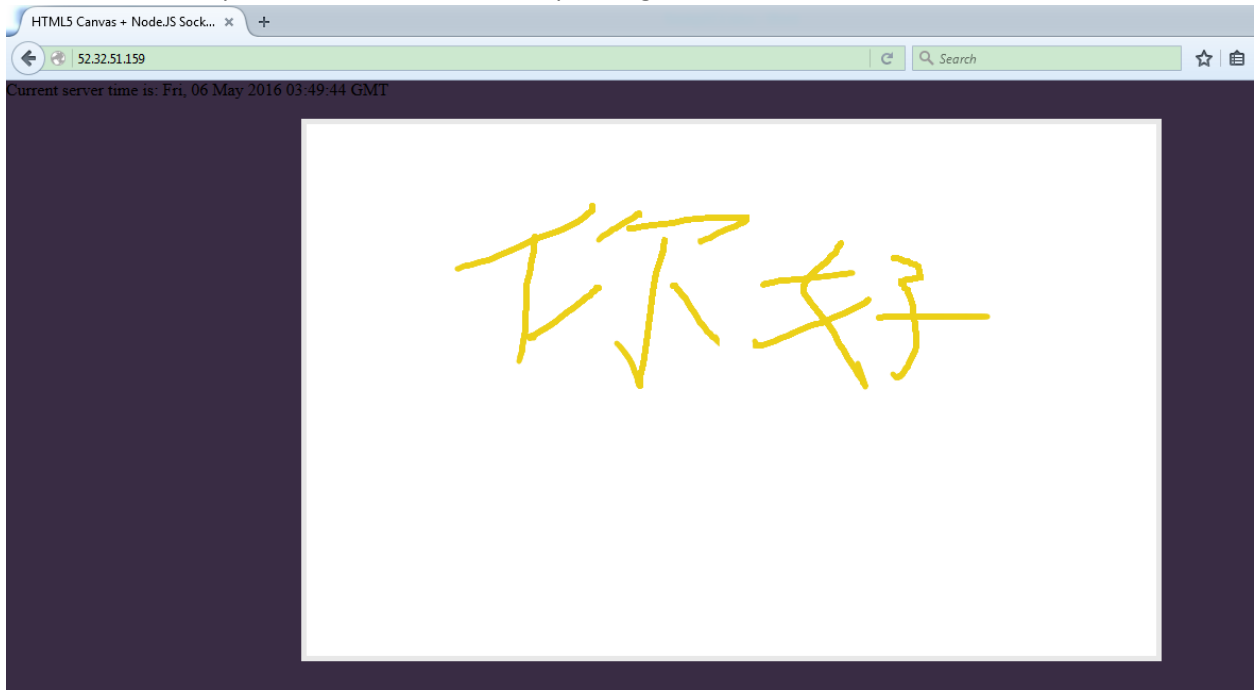
5.4 Open Chrome, Firefox, on different device.

This node project is a drawing application. It uses Socket.IO to broadcast the changes from one client to other clients.

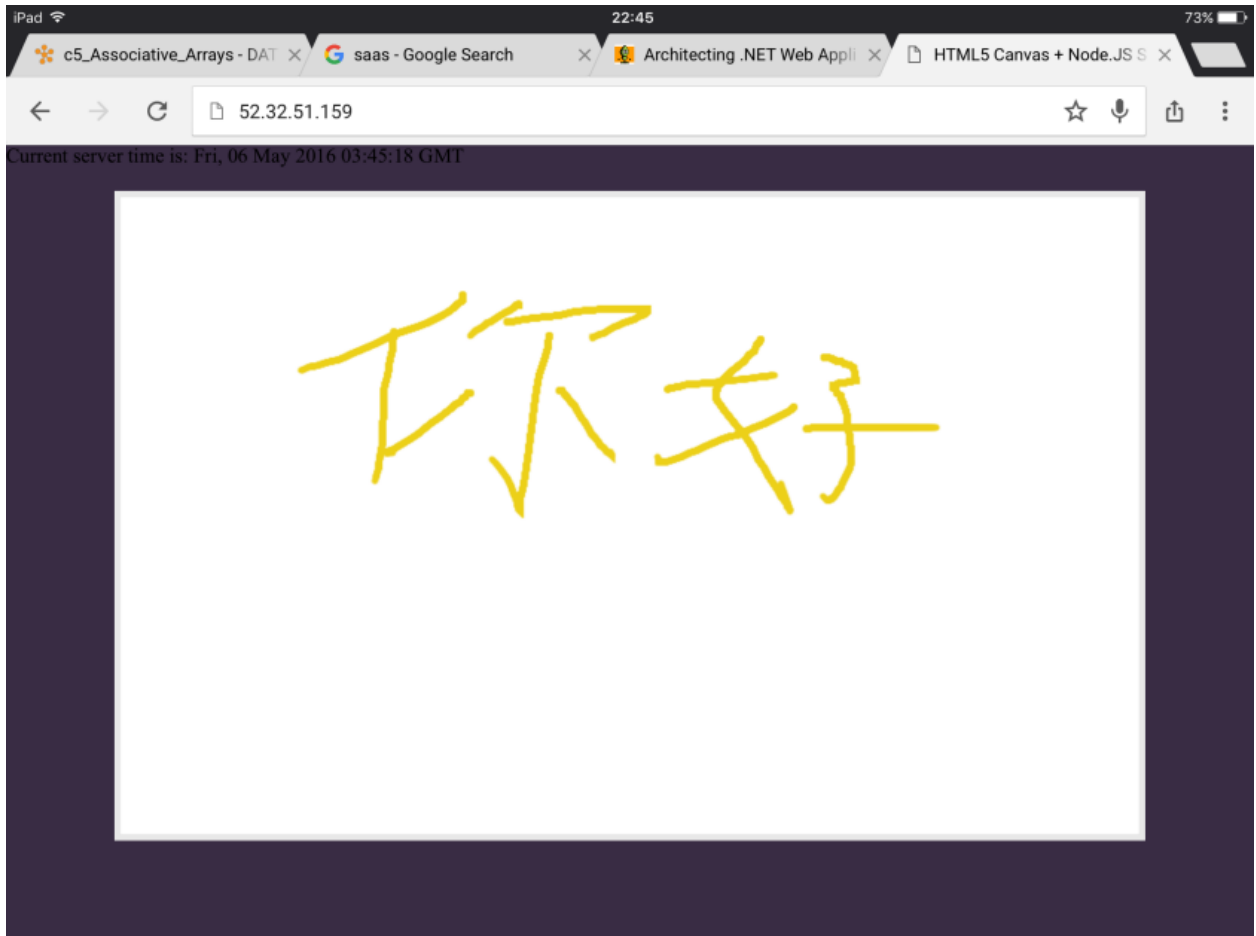
The first user opens it in chrome, waits others to join and draws something later.



The second user opens it in Firefox. The same painting as the first user draws.



The third user open it on iPad.



The forth user opens it on iPhone.



6. Useful command in linux

Command	Description
ls	Show files/directories under the current folder
sudo su	Switch to root user
sudo su -- ec2-user	Switch to ec2-user
nano filename	Open/Create file with nano
mkdir foldername	Create folder
sudo make uninstall	Uninstall, go to the folder and run it.

7. Reference

[How I Got Node.js Running On A Linux Micro Instance Using Amazon EC2](#)

<http://www.bennadel.com/blog/2321-how-i-got-node-js-running-on-a-linux-micro-instance-using-amazon-ec2.htm>

How to install & setup Node.js on Amazon EC2 – complete guide

<http://iconof.com/blog/how-to-install-setup-node-js-on-amazon-aws-ec2-complete-guide/>

Setup Node.js Environment on Amazon EC2 linux

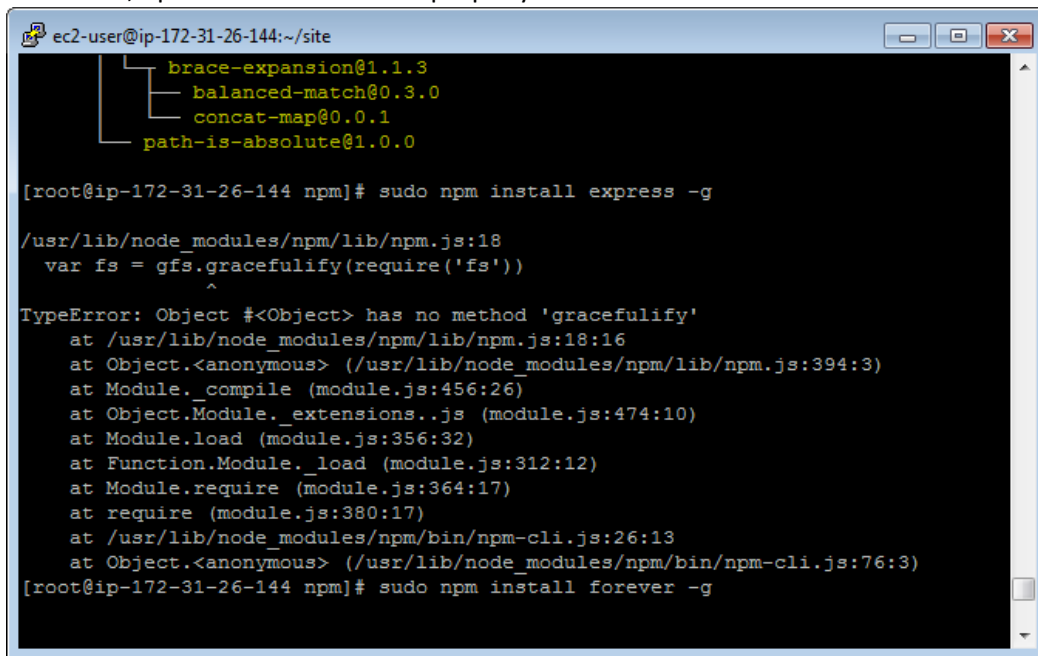
http://adndevblog.typepad.com/cloud_and_mobile/2014/12/setup-nodejs-environment-on-amazon-ec2-linux-1.html

<https://docs.npmjs.com/misc/removing-npm>

8. Issues

8.1 Remove npm

Sometime, npm itself doesn't work properly.

A terminal window titled 'ec2-user@ip-172-31-26-144:~/site' showing a tree view of installed packages: brace-expansion@1.1.3, balanced-match@0.3.0, concat-map@0.0.1, and path-is-absolute@1.0.0. Below the tree, the user runs 'sudo npm install express -g'. The terminal shows the start of the installation process, including the file path '/usr/lib/node_modules/npm/lib/npm.js:18' and the code 'var fs = gfs.gracefulify(require('fs'))'. A 'TypeError' is thrown: 'Object #<Object> has no method 'gracefulify''. The stack trace includes: 'at /usr/lib/node_modules/npm/lib/npm.js:18:16', 'at Object.<anonymous> (/usr/lib/node_modules/npm/lib/npm.js:394:3)', 'at Module.compile (module.js:456:26)', 'at Object.Module._extensions..js (module.js:474:10)', 'at Module.load (module.js:356:32)', 'at Function.Module._load (module.js:312:12)', 'at Module.require (module.js:364:17)', 'at require (module.js:380:17)', 'at /usr/lib/node_modules/npm/bin/npm-cli.js:26:13', and 'at Object.<anonymous> (/usr/lib/node_modules/npm/bin/npm-cli.js:76:3)'. The prompt returns to '[root@ip-172-31-26-144 npm]# sudo npm install forever -g'.

Then we have to uninstall and install it again.

`sudo npm uninstall npm -g`

If it doesn't work, go the 'npm' folder, run:

`sudo make uninstall`