

CS 360

How to install & setup Node.js on Amazon EC2

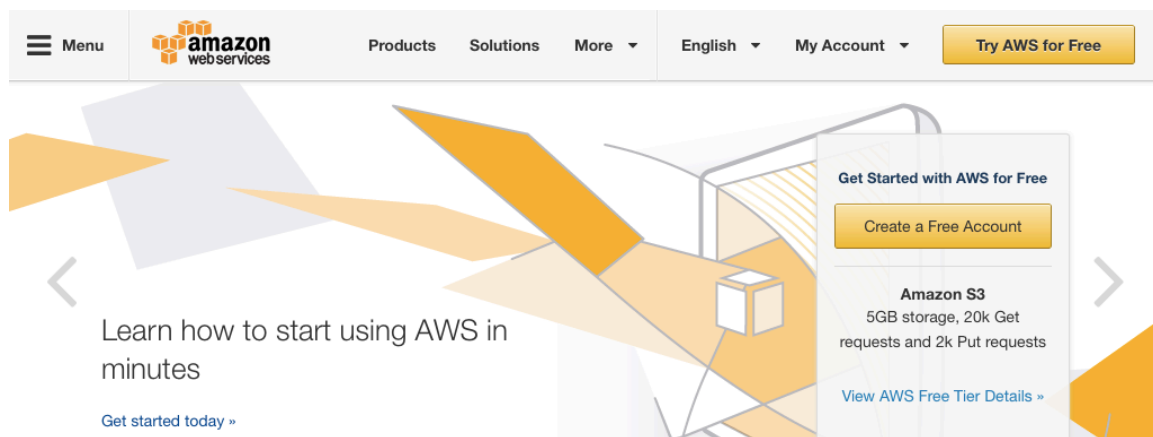
You can use the following steps to create a web server that you can use for the Node.js labs for the rest of the semester. You will be turning in a URL that will access your application on your EC2 server.

During this process you will become the administrator of a Linux machine and will have superuser permissions. Along with learning how to build your web application, you ought to spend some time learning system administration concepts. So, if things seem unfamiliar, ask questions in class.

The following steps should get you up and running. Try to follow them closely, then you can come back later and explore.

Create an Amazon Web Services (AWS) account

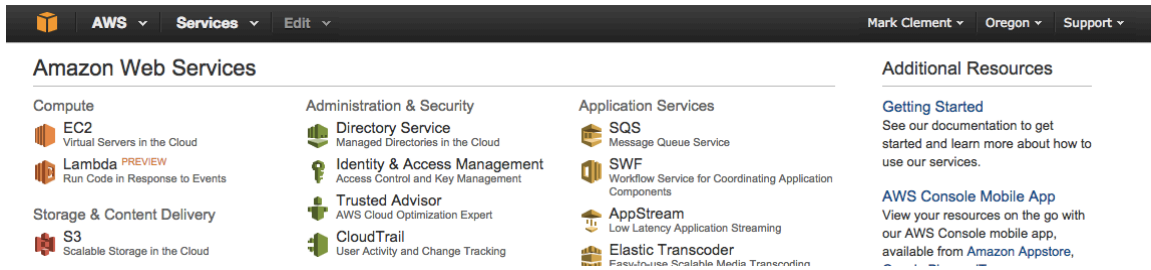
Create an [Amazon Web Services \(AWS\) account](#)



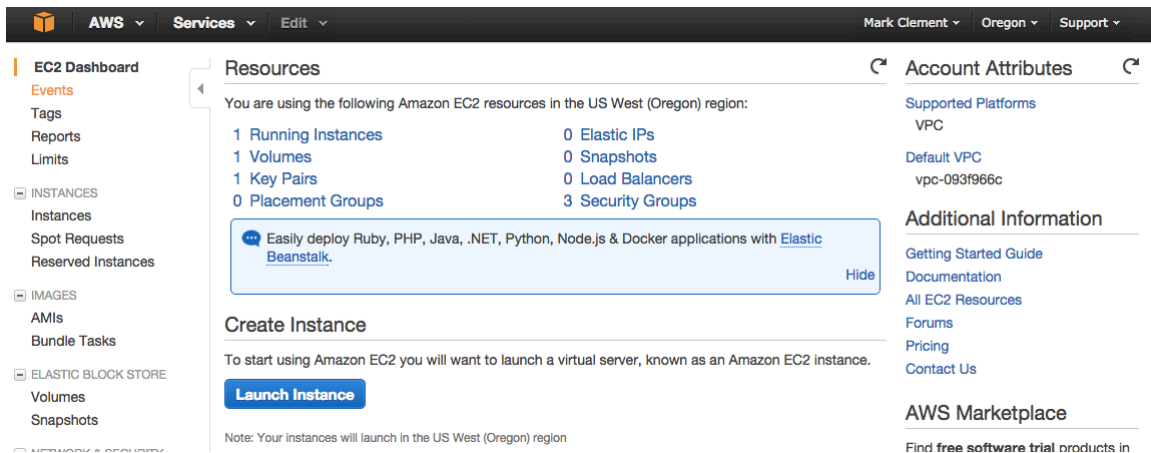
Launch an EC2 instance (web server)

In order to install and run Node, we first need to have a web server. Amazon calls its web servers *instances*, because you can have many of them running in parallel. For now, you only need one instance.

Login with your AWS account and goto the EC2 button in the upper left hand corner.

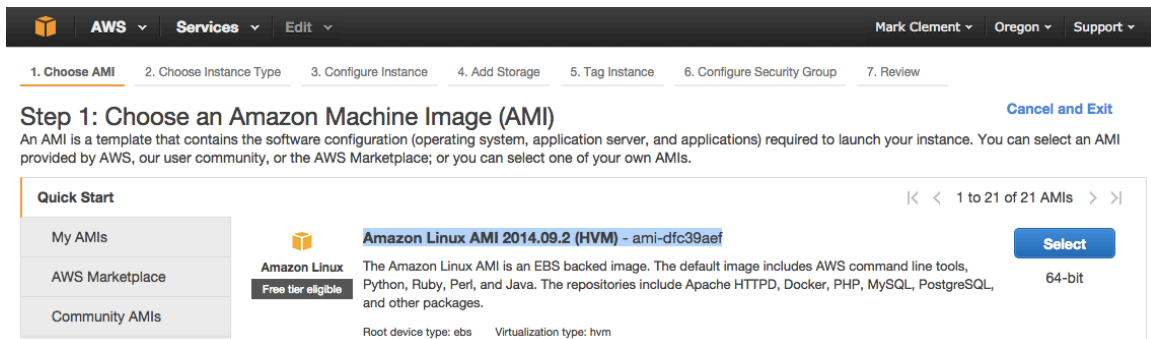


Then select the button to launch an instance

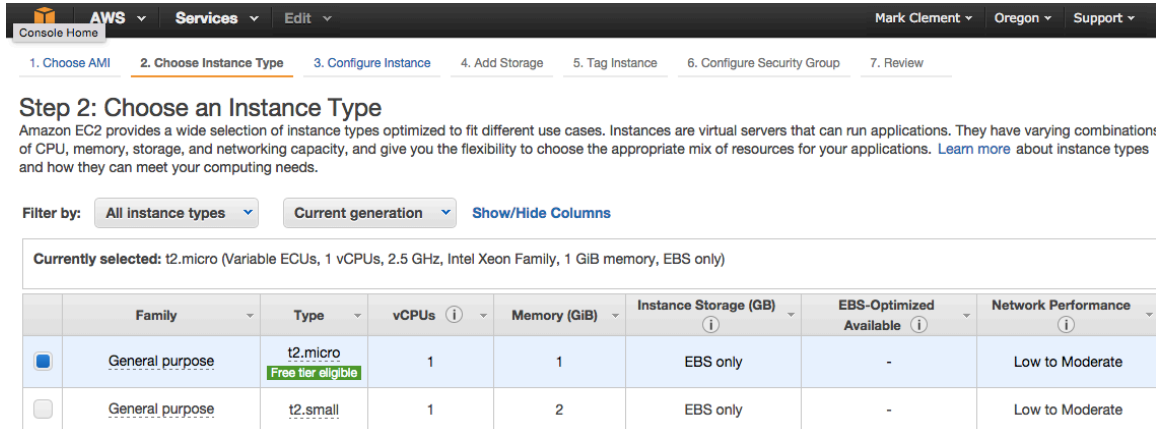


Step 1: Choose an Amazon Machine Image (AMI)

Make sure you select the free tier instances. I chose the **Amazon Linux AMI 2014.09.2 (HVM) - ami-dfc39aef**



Once you select this instance, select the t2.micro since it is the only free tier option. And select the “Next: Configure Instance Details” button in the bottom right corner of the screen.



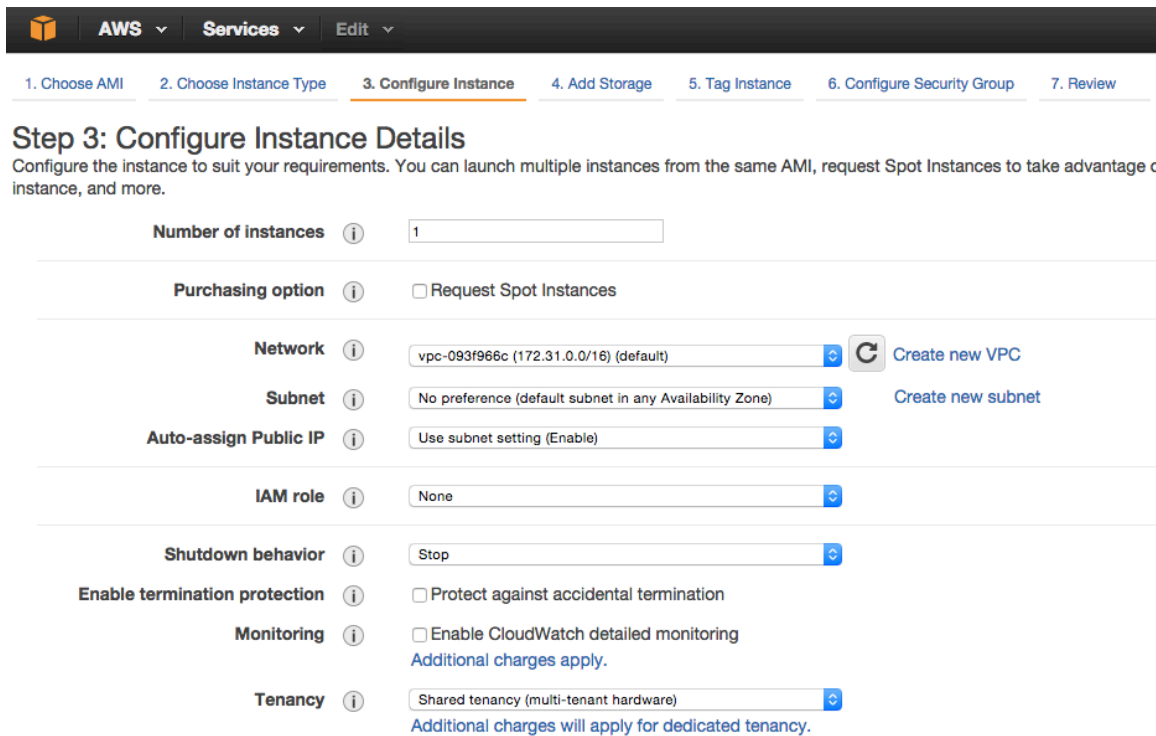
Step 2: Choose an Instance Type
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose 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 checked="" type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate

You can use the defaults for step 3 and proceed to “Next: Add Storage”



Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of a Spot instance, and more.

Number of instances

Purchasing option Request Spot Instances

Network [Create new VPC](#)

Subnet [Create new subnet](#)

Auto-assign Public IP

IAM role

Shutdown behavior

Enable termination protection Protect against accidental termination

Monitoring Enable CloudWatch detailed monitoring
[Additional charges apply.](#)

Tenancy
[Additional charges will apply for dedicated tenancy.](#)

Use the default storage parameters and proceed to “Next: Tag Instance”

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Delete on Termination	Encrypted
Root	/dev/xvda	snap-f518b274	8	Magnetic	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Here you will select a name value pair for your instance. I just used the suggested “Name” and “Webserver” tag.

Step 5: Tag Instance

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#)

Key (127 characters maximum)	Value (255 characters maximum)
Name	Webserver

Select the “Next: Configure Security Group” button. I enabled ssh, http, https and a block of 10 ports from 3000 - 3010 for use in starting new web services (this is where node.js will run by default).

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group
 Select an existing security group

Security group name: launch-wizard-3
Description: launch-wizard-3 created 2015-02-16T08:58:28.643-07:00

Type	Protocol	Port Range	Source
SSH	TCP	22	Anywhere 0.0.0.0/0
HTTP	TCP	80	Anywhere 0.0.0.0/0
HTTPS	TCP	443	Anywhere 0.0.0.0/0
Custom TCP Rule	TCP	6000-6010	Anywhere 0.0.0.0/0

Now select “Review and Launch” in the bottom right corner and select the default boot volume.

Boot from General Purpose (SSD) ✕

General Purpose (SSD) volumes provide the ability to burst to 3,000 IOPS per volume, independent of volume size, to meet the performance needs of most applications and also deliver a consistent baseline of 3 IOPS/GiB.

- Make General Purpose (SSD) the default boot volume for all instance launches from the console going forward (recommended).
- Make General Purpose (SSD) the boot volume for this instance.
- Continue with Magnetic as the boot volume for this instance.



Free tier eligible customers can get up to 30GB of General Purpose (SSD) storage.

Next

Once you have clicked “Next” , select “Launch” in the bottom right corner. You will then create a public key pair that you will use to ssh into the virtual sever. Select “create a new key pair” and give it a name, then select “Download Key Pair” . This should download a file with a .pem suffix. Keep track of this file, so you can use it in the future. Now select “Launch Instances”

Select an existing key pair or create a new key pair



A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair



Key pair name

clement

Download Key Pair



You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel

Launch Instances

Your instance is now launching and may take a few minutes. You can select “View Instances” to see how things are going.

AWS Services Edit Mark Clement Oregon Support

Launch Status

Your instances are now launching
The following instance launches have been initiated: [i-c6bda9cc](#) [View launch log](#)

Get notified of estimated charges
[Create billing alerts](#) to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. [Find out](#) how to connect to your instances.

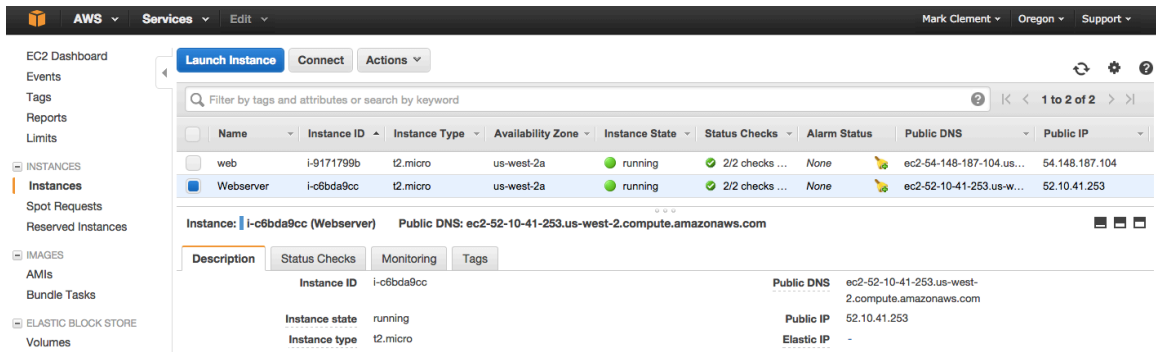
Here are some helpful resources to get you started

- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2: User Guide
- Amazon EC2: Discussion Forum

While your instances are launching you can also

- [Create status check alarms](#) to be notified when these instances fail status checks. (Additional charges may apply)
- [Create and attach additional EBS volumes](#) (Additional charges may apply)
- [Manage security groups](#)

[View Instances](#)



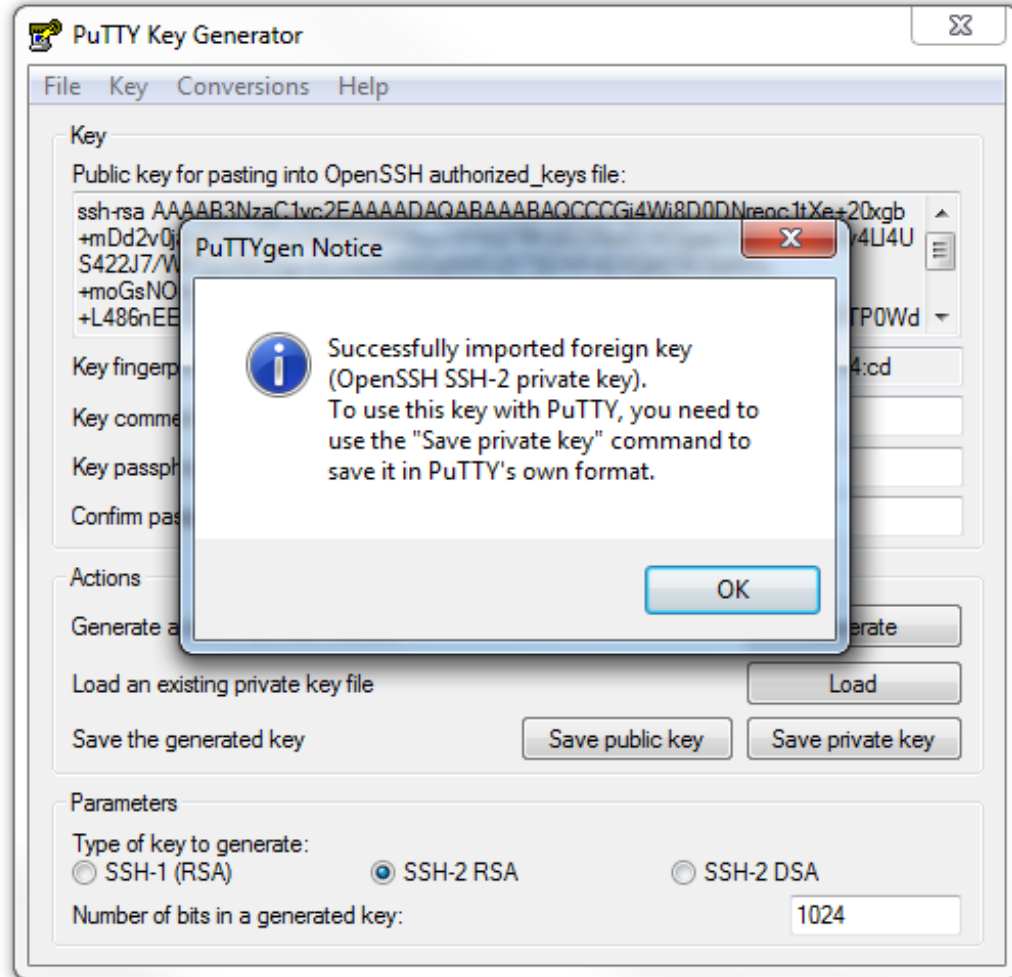
You can now use your instance's public dns to connect to your server via ssh. Amazon does not provide a custom username and password for ssh connections. Instead, they use the key pair file you created a couple of steps ago. Note the public DNS and public IP addresses that are shown in this screen. You will need them to connect to your server

Connect to Amazon EC2 with Putty

If you're on Windows, you can use an ssh client like [Putty](#):

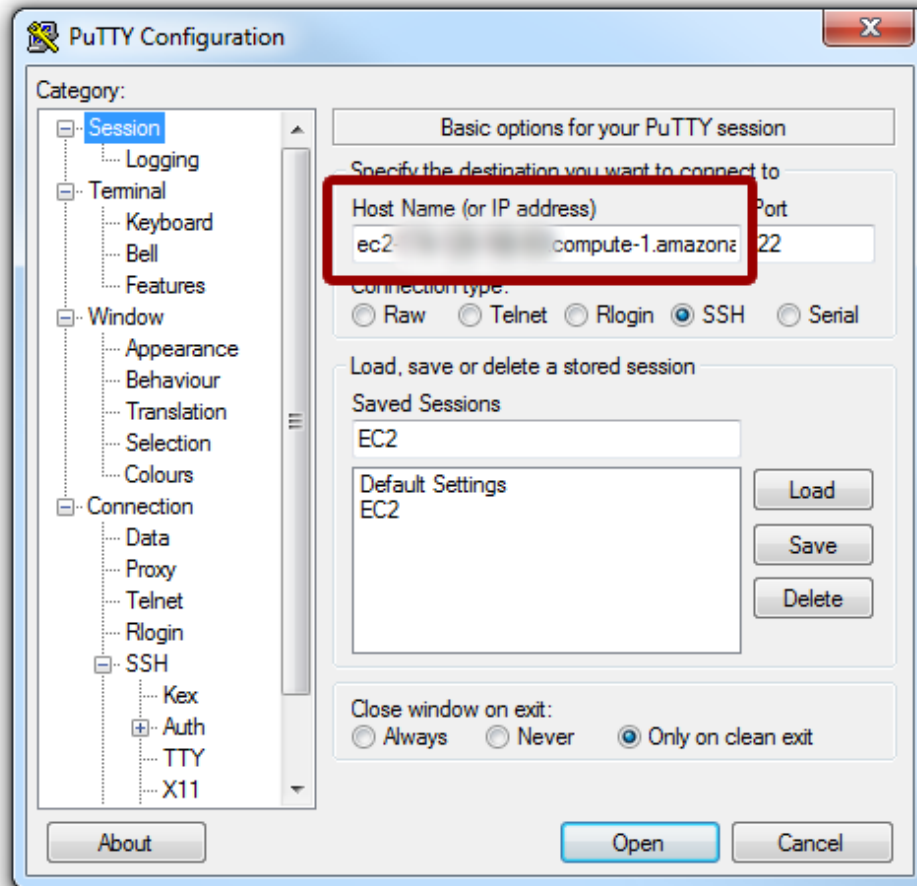
- [Download Putty and puttygen](#)
- Use puttygen to convert Amazon's .pem key pair file to .ppk file.
 - Start puttygen and select **Load**

- Select *view all files* and pick the .pem file you download from Amazon.

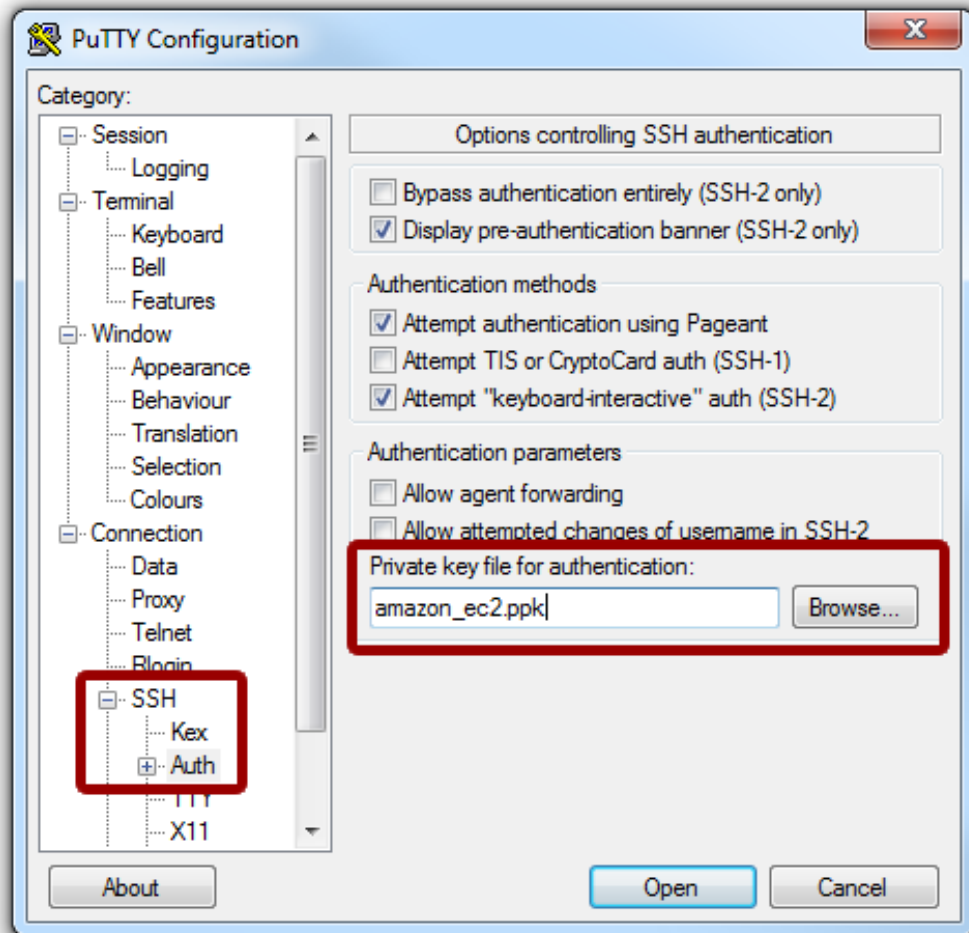


- Click **OK** and select **Save Private Key**. A passphrase is not required but recommended for additional security.
- Connect with Putty.

- Launch Putty and enter your instance's public dns address.

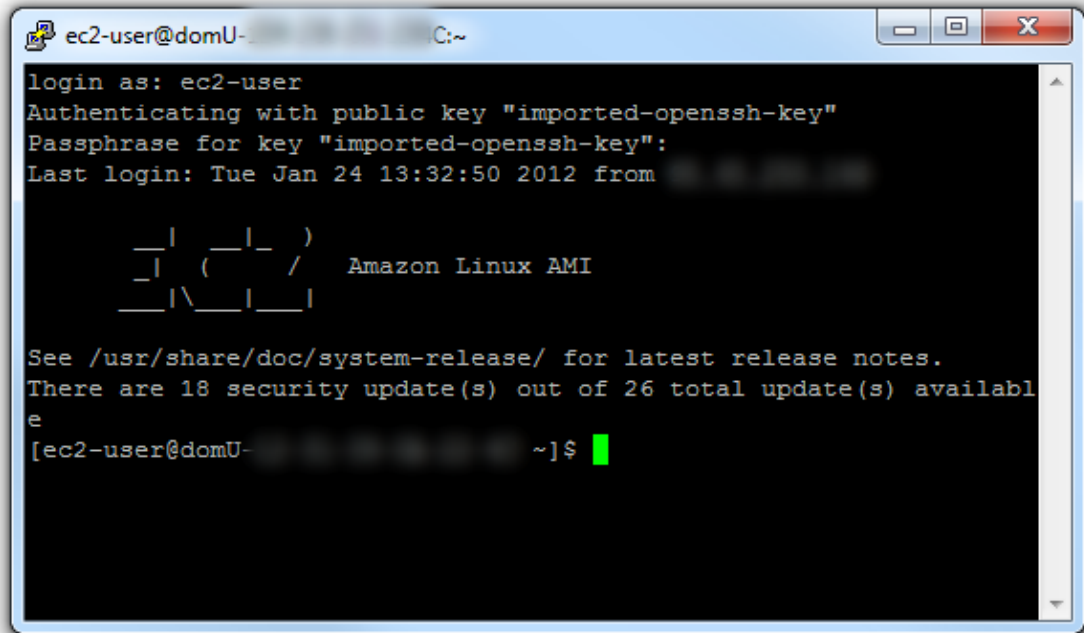


- Navigate to Connection/SSH/Auth. Click **Browse** and select the .ppk file you exported from puttygen.



- Finally, click **Open**. When the connection comes up for the first time, Putty will ask you if you want to save the server's credentials; click **Yes**. In the *login as* prompt, type *ec2-user* and then your passphrase key for

your key pair (if any). You are now logged-in into your instance!



Apple or Linux

You will first need to change the permissions on your .pem file.

```
chmod 600 ~/Downloads/clement.pem
```

If you are using an apple or linux machine, you will use ssh instead of putty.

Open a terminal window and run ssh with a command like:

```
ssh -i ~/Downloads/clement.pem ec2-user@52.10.41.253
```



```
sudo ./configure
sudo make
sudo make install
```

Alright, Node is now installed! Let's add it to sudo's path so that we can install more packages. You will need to use the VI editor to edit the `/etc/sudoers` file.

Type the following:

```
sudo su
vi /etc/sudoers
```

If you're not familiar with VI, do the following:

```
<Use the down keyboard arrow to find this line:>
Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin

<Use the right arrow to move the cursor to the end of the line and press the INSERT button. Now type>
:/usr/local/bin
<At this point the line should be>
Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin
<In order to save your changes and exit VI, hit ESC and type>
:

<now type>
wq!

<and you're back in the console>
<in order to leave su mode, type:>
exit
```

The next step is to install [NPM](#)(Node package manager). Type the following commands:

```
cd /usr/local/src
sudo git clone https://github.com/isaacs/npm.git
cd npm
sudo make install
```

You now have a working Amazon EC2 instance with Node.js and NPM! You can install additional Node packages using NPM:

```
sudo npm install express -g
sudo npm install forever -g
sudo npm install connect serve-static -g
```

Write a simple web server in node.js

First, create a directory that you can work in and use `chmod` to make it writable by your normal `ec2-user`.

```
sudo mkdir /var/node
sudo chown ec2-user /var/node
```

Now, create a file with vim or emacs with the following content:

```
var http = require('http');
var messages = [
  'Hello World',
  'From a basic Node.js server',
  'Take Luck'];
http.createServer(function (req, res) {
  res.setHeader("Content-Type", "text/html");
  res.writeHead(200);
  res.write('<html><head><title>Simple HTTP Server</title></head>');
  res.write('<body>');
  for(var idx in messages) {
    res.write('\n<h1>'+messages[idx]+'</h1>');
  }
  res.end('\n</body></html>');
}).listen(80);
```

Save it out as simple.js. And run it with sudo so that you can use port 80.

```
sudo node simple.js
```

You should be able to access this server through the IP address

```
http://52.10.41.253/
```

in a web browser.