Tutorial: Setting Up NGINX and NGINX Plus as a Web Server and Reverse Proxy in AWS

Since its introduction more than a decade ago, NGINX has become the most popular web server for busy websites. More than 160 million websites use NGINX, including [more than half of the top 100,000 websites](https://w3techs.com/technologies/cross/web_server/ranking).

With NGINX having reached such popularity, it makes sense to use NGINX for all your [web‑serving](https://www.nginx.com/resources/glossary/web-server/) needs, and additionally to take advantage of its capabilities as a [reverse proxy server](https://www.nginx.com/resources/glossary/reverse-proxy/), [caching server](https://www.nginx.com/resources/glossary/caching/), [load balancer](https://www.nginx.com/resources/glossary/load-balancing/), and more.

This blog post guides you in getting NGINX and NGINX Plus up and running on Ubuntu 16.04 on Amazon Web Services (AWS), an easy‑to‑use setup that doesn’t affect the settings on your personal computer. This blog post covers both NGINX and NGINX Plus.

Once you complete the steps given here, you may want to continue with the [NGINX Core](https://university.nginx.com/instructor-led-training/nginx-core2) training, a full‑day course where you’ll learn how to configure, administer, and manage NGINX. The course also readies you for our other instructor‑led and self‑paced [courses on advanced topics](https://university.nginx.com/).

Accessing a Terminal

For this tutorial, you need a terminal to connect to your Amazon Elastic Compute Cloud (EC2) instance. Macs have a built‑in terminal, but Windows users need to download PuTTY or an equivalent tool. If you prefer to use your own local terminal to set up NGINX or NGINX Plus and serve web content, feel free to skip to [Setting up AWS](https://www.nginx.com/blog/setting-up-nginx/#aws-setup).

Accessing a Terminal on MacOS

On Macs, you can use the built‑in terminal tool to do everything in this tutorial.

1. Click the magnifying glass icon at the right end of the top bar on the screen.
2. Type **terminal** in the pop‑up box and press Enter. A separate terminal window opens.

Accessing a Terminal on Windows

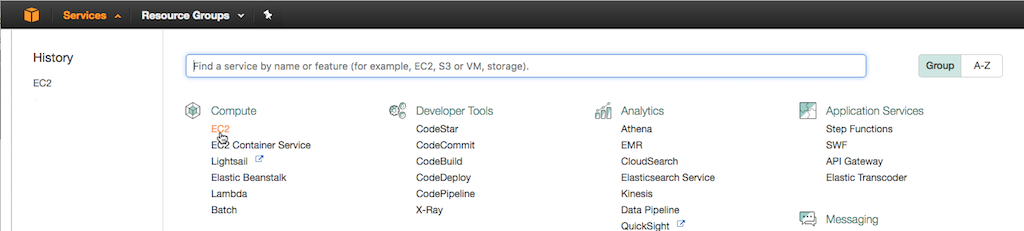
Windows doesn’t come with a built‑in terminal, so you will need to download one from the internet. PuTTY is a common ssh client used ubiquitously throughout the world.

1. Access <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>.
2. Download and install the file in a location of your choice, then open it. You can’t access the actual terminal until you finish the steps in the [next section](https://www.nginx.com/blog/setting-up-nginx/#aws-setup), however.

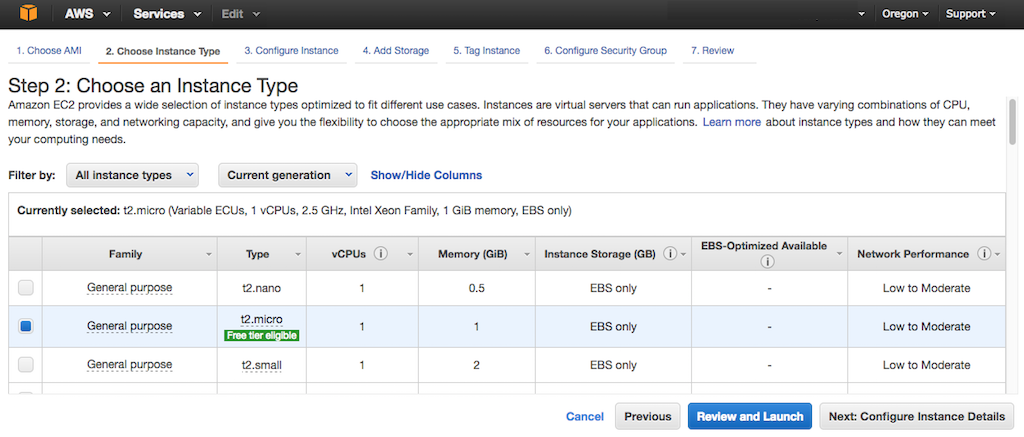
Setting Up AWS

Set up your AWS deployment by creating an EC2 instance and connecting to it.

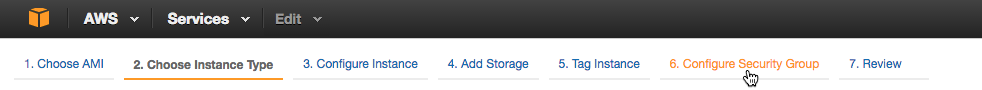
1. On the [AWS home page](https://aws.amazon.com/), sign in to the (AWS Management) Console or create a new account.
2. Click **Services** in the top Console navigation bar, then **EC2** in the **Compute** section.



1. Click the  Launch Instance  button on the page that opens.
2. On the **Step 1** page, click the  Select  button In the **Ubuntu Server 16.04 LTS (HVM), SSD Volume Type** row.
3. On the **Step 2** page, select the **t2.micro** instance type, which as of this writing is marked  Free tier eligible  and selected by default.

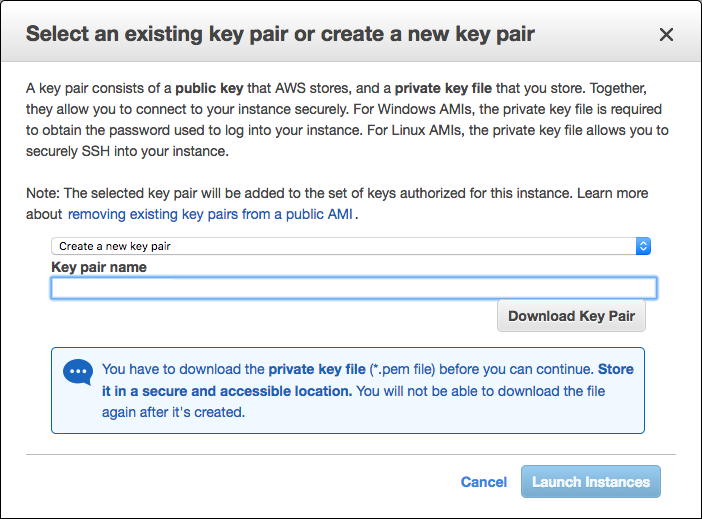


1. By default EC2 instances accept only SSH traffic. To allow incoming HTTP and HTTPS traffic:
   1. At the top of the window, click 6. Configure Security Group.

[](https://www.nginx.com/wp-content/uploads/2016/10/aws-ec2-configure-security-group.png)

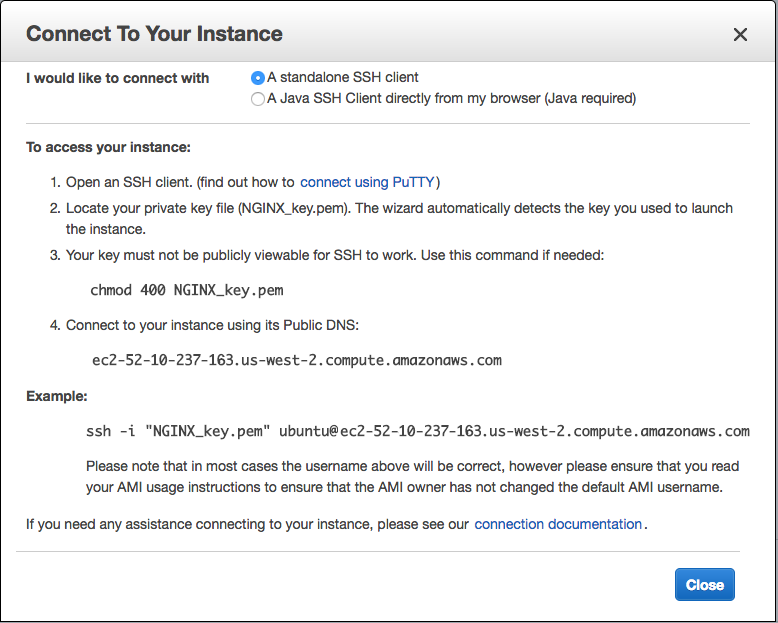
* 1. On the **Step 6** page, click the **Add Rule** button below the table, then select **HTTP** in the **Type** column. We are allowing access to our instances from any IP address (the value in the **Source** column is Anywhere).
  2. Repeat Step b for **HTTPS**.

1. Click the  Review and Launch  button at the bottom of the page.
2. On the **Step 7** page, click the  Launch  button at the bottom.
3. In the box that pops up, create a new key pair:
   1. Select **Create a new key pair** from the upper drop‑down menu.
   2. Type a name in the **Key pair name** field, such as **NGINX\_key**.

[](https://www.nginx.com/wp-content/uploads/2016/10/aws-ec2-key-pair-name.png)

* 1. Click the **Download Key Pair** button.
  2. In your file manager utility, move the downloaded **.pem** file to a secure location. For the tutorial we’re placing it in the **/Desktop/NGINX** directory.
  3. Click the  Launch Instances  button.

1. On the **Launch Status** page that opens, click the  View Instances  button at the bottom.
2. On the page that opens, the new instance appears in the table. Give it a name:
   1. Click the pencil icon in the **Name** column.
   2. Type the name in the field and click the check‑mark icon.
3. Click the **Connect** button at the top of the screen. A window like the following pops up.

[](https://www.nginx.com/wp-content/uploads/2016/10/aws-ec2-connect-instance-popup.png)

1. Follow the directions on the pop‑up to finish connecting to your instance (including the ones accessible by clicking **connect using PuTTY**, if you’re using Windows). This includes pasting the sample ssh command into your terminal once you’ve navigated to your key.

Installing NGINX Software

Now that you have your AWS virtual machine (EC2 instance) set up, it’s time to install either the [open source NGINX software](https://www.nginx.com/blog/setting-up-nginx/#install-open-source) or [NGINX Plus](https://www.nginx.com/blog/setting-up-nginx/#install-nginx-plus), which is available for free in a 30‑day trial). Both options work in the context of this tutorial, but if you want to further explore the advanced features in NGINX Plus, please request a free trial.

Installing Open Source NGINX

To install the open source NGINX software, follow these steps:

1. [Access your terminal.](https://www.nginx.com/blog/setting-up-nginx/#terminal)
2. Download the NGINX signing key:

**sudo wget http://nginx.org/keys/nginx\_signing.key**

1. Add the key:

**sudo apt-key add nginx\_signing.key**

1. Change directory to **/etc/apt**.

**cd /etc/apt**

1. Edit the **sources.list** file, appending this text at the end:
2. deb http://nginx.org/packages/ubuntu xenial nginx

deb-src http://nginx.org/packages/ubuntu xenial nginx

1. Update the NGINX software:

**sudo apt-get update**

1. Install NGINX:

**sudo apt-get install nginx**

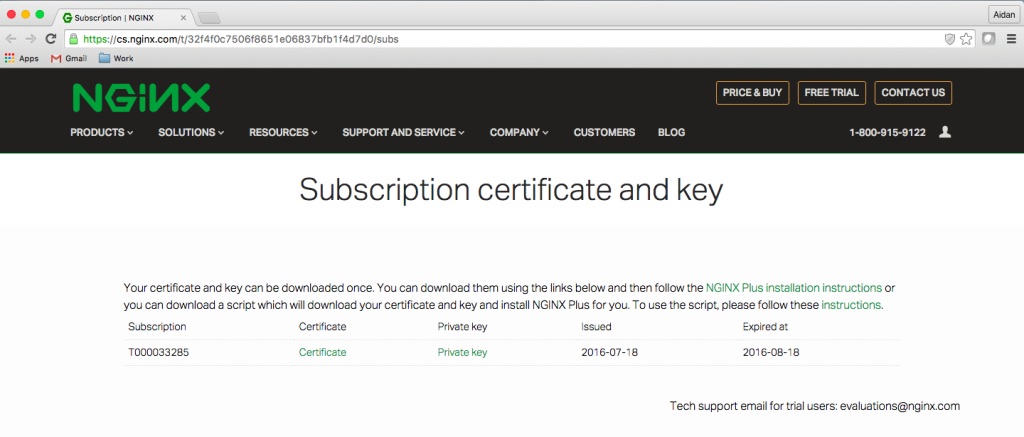
1. Type **Y** when prompted.
2. Start NGINX:

**sudo service nginx start**

1. Continue to [Opening Your Web Page](https://www.nginx.com/blog/setting-up-nginx/#open-web-page).

Installing NGINX Plus

1. If you don’t already have NGINX Plus, then [sign up for a 30‑day free trial](https://www.nginx.com/blog/setting-up-nginx/#free-trial).
2. When you receive notification that your subscription is available, log in at the [NGINX Plus Customer Portal](https://cs.nginx.com/). Access the installation instructions by clicking either of the hyperlinks shown in the screenshot.

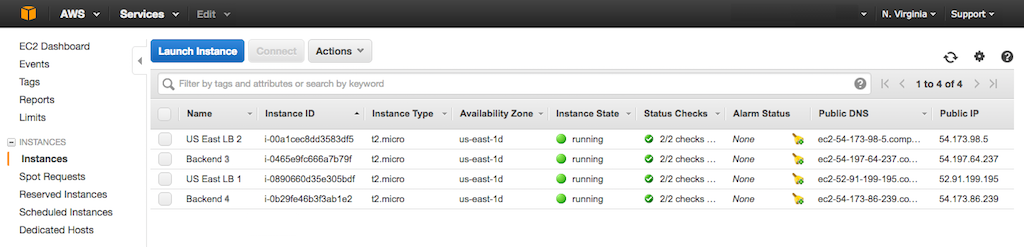


1. When installation is complete and NGINX Plus is running, continue to [Opening Your Web Page](https://www.nginx.com/blog/setting-up-nginx/#open-web-page).

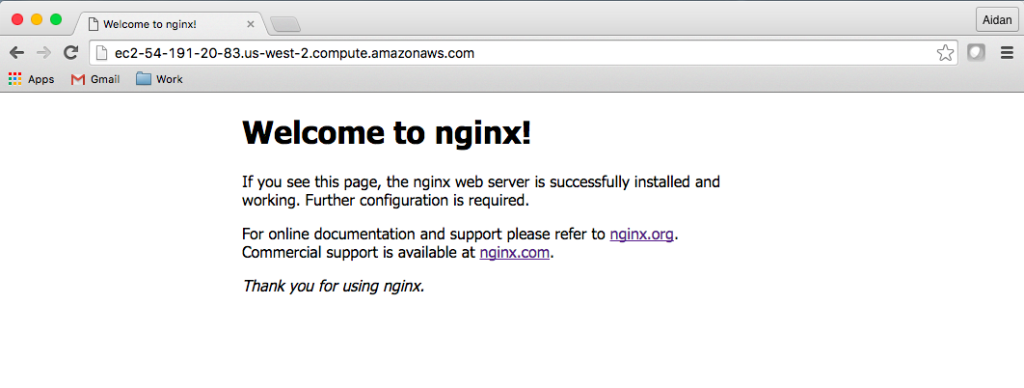
Opening Your Web Page

Now that we’ve started the NGINX software, we’ll look at the web page that NGINX and NGINX Plus serve by default before you configure it to deliver your site’s content. Follow these steps:

1. Navigate to the **Instances** tab on the EC2 Dashboard if you are not there already (click **Instances** in the left‑hand column.)

[](https://www.nginx.com/wp-content/uploads/2016/10/aws-ec2-useast-instances.png)

1. Select the instance’s IP address in the **Public DNS Address** column and copy it into the paste buffer.
2. Open a new tab in the browser and paste the address into the address bar. This appears in the window:



Setting Up Sample Files

Now that we know we have a working version of NGINX or NGINX Plus installed, it’s time to put it to good use! Let’s begin by setting up some files and directories.

1. Change directory to your home directory if you are not already there. In the following instructions, it is **/home/ubuntu**.
2. Create a folder called **public\_html** and change into it.
3. Inside the new folder, create a file called **index.html** and a folder called **application1**.
4. In the **application1** folder, create a file called **app1.html** with some text in it.
5. Change back to your home directory.
6. Create a folder called **data**.
7. Within the **data** folder, create a folder called **images**.

Serving Pages and Images

Our first use case for NGINX or NGINX Plus is to serve pages and images to the user via our web page.

1. Change directory to **/etc/nginx/conf.d**.
2. Rename **default.conf** to **default.conf.bak** to prevent NGINX or NGINX Plus from using it as the default configuration file.
3. Create a file called **server1.conf** with this configuration in it:
4. server {
5. root /home/ubuntu/public\_html;
6. location /application1 { }
7. location /images {
8. root /home/ubuntu/data;
9. }

}

1. Change directory to **~/data/images**.
2. Find an image that you want to serve and copy it to the **~/data/images** directory. As an example, the following command copies in the [NGINX logo](https://cdn.wp.nginx.com/wp-content/themes/nginx-theme/assets/img/logo.png):

**sudo wget https://cdn.wp.nginx.com/wp-content/themes/nginx-theme/assets/img//logo.png**

1. Reload NGINX or NGINX Plus:

**sudo nginx -s reload**

1. Open your web page and request the image at this URL:

**https://*NGINX-server*/images/logo.png**

1. Also access the application and observe what you get:

**https://*NGINX-server*/application1/app1.html**

Setting Up a Proxy Server

Now that we have a working web server, it’s time to learn how to configure it to route traffic. This capability enables you to pass traffic through to other servers and is a major step towards setting up load balancing. Follow these steps:

1. In the **~/data** folder, create a folder called **server2**.
2. In the **server2** folder, create a folder called **sampleApp**.
3. Inside the **sampleApp** folder, create a file named **index.html** and write some text in it.
4. Change directory to **/etc/nginx/conf.d**.
5. Create a file called **server2.conf** with this configuration in it:
6. server {
7. listen 8080;
8. root /home/ubuntu/data/server2;

}

1. Modify **server1.conf** as follows:
2. server {
3. root /home/ubuntu/public\_html;
4. location /application1 {
5. proxy\_pass http://localhost:8080/sampleApp;
6. }
7. location /images {
8. root /home/ubuntu/data;
9. }

}

1. Reload NGINX or NGINX Plus:

**sudo nginx -s reload**

1. Access this URL in your browser and observe what has changed compared to when you accessed it in [Serving Pages and Images](https://www.nginx.com/blog/setting-up-nginx/#web-server):

**https://*NGINX-server*/application1/index.html**

So that’s it! You now have a working Ubuntu instance running NGINX, which is ready to run as a proxy server.

Conclusion

In this post, we learned how to serve files and images over the Internet, as well as configure it as a reverse proxy. If you have any questions about the tutorial, we offer an [NGINX Core](https://university.nginx.com/instructor-led-training/nginx-core2) training course that covers this material in much more depth. The course also covers some of the essentials in web‑serving and application delivery, such as load balancing, location routing, and security. You will have a chance to speak to an expert in the field about any questions you have, as well as delve deeper into the power of NGINX and NGINX Plus.