**Deploying Node JS Application With Nginx On Ubuntu**

**Installing Node JS Using PPA**

A way that can get you the more recent version of Node.js is to add a PPA (Personal Package Archive) maintained by NodeSource. This will have more up-to-date versions of Node.js than the official Ubuntu repositories, and allows you to choose between Node.js v4.x (the older long-term support version, which will be supported until April of 2018), Node.js v6.x (supported until April of 2019), and Node.js v8.x (the current LTS version, supported until December of 2019).

First, you need to install the PPA in order to get access to its contents. Make sure you're in your home directory, and use curl to retrieve the installation script for your preferred version, making sure to replace 8.x with your preferred version string (if different):

|  |
| --- |
| $ cd ~ $ curl -sL https://deb.nodesource.com/setup\_8.x -o nodesource\_setup.sh |

And run the script under sudo:

|  |
| --- |
| $ sudo bash nodesource\_setup.sh |

The PPA will be added to your configuration and your local package cache will be updated automatically. After running the setup script from nodesource, you can install the Node.js package in the same way you did above:

|  |
| --- |
| $ sudo apt-get install -y nodejs |

To check the version of Node.js you have installed, type:

|  |
| --- |
| $ nodejs -v  Output:  V8.10.0  $ npm -v  Output:  5.6.0 |

**Create a Node JS Application**

We will create a simple application that returns "Hello World" to any incoming HTTP requests. This is merely a sample application that's going to help you get Node.js set up. You can replace it with your own application, and modify the application to listen on the correct IP addresses and ports.

|  |
| --- |
| $ cd ~  $ vim server.js |

Insert the code below into your server.js file:

|  |
| --- |
| var http = require('http');  http.createServer(function (req, res){  res.writeHead(200, {'Content-Type': 'text/plain'});  res.end('Hello World\n');  }).listen(8080, 'APP\_SERVER\_PUBLIC\_IP');  console.log('Server running at http://APP\_SERVER\_PUBLIC\_IP:8080/'); |

This Node.js application will listen on the specified port (8080) and IP addresses and return "Hello World" with a 200 HTTP success code.

**Test the application (optional)**

Test your application on your app server:

|  |
| --- |
| $ node server.js |

Test your application by opening [http://APP\_SERVER\_PUBLIC\_IP:8080](http://app_server_public_ip:8080) in a browser window, using your public IP. If you receive the "Hello World" output, your application is working properly. If you received an incorrect output, make sure your Node.js application is both running and also configured to listen on the correct IP address and port.

**Install and Configure PM2**

We will use PM2 to manage Node.js applications. PM2 makes it easy to automate and manage applications by running them as a service. Use Node.js Package Manager (NPM) to install PM2 on your app server.

|  |
| --- |
| $ sudo npm install -g pm2 |

Start application with PM2 so that it runs in the background:

|  |
| --- |
| $ pm2 start server.js |

Your application will be added to PM2's process list, which will be displayed each time you start an application.

**Setting Up Reverse Proxy With Nginx**

Now that we have established that your Node.js application is running and listening on the Private address, we now need a way for users to access it on the web. To do this, we will set up an NGINX web server as a reverse proxy, which will allow access for our users.

**Update your packages and install Nginx**

|  |
| --- |
| $ sudo apt-get update  $ sudo apt-get install nginx |

**Create an Nginx configuration file and add a server block**

|  |
| --- |
| $ sudo vi /etc/nginx/sites-enabled/server\_block.conf |

Add the following lines into the *server\_block.conf* file you just created:

|  |
| --- |
| server {  listen 80;  listen [::]:80;  server\_name <your\_domain>; #Ex.: dev.smartories.com  location / {  proxy\_pass http://APP\_SERVER\_PUBLIC\_IP:8080  }  } |

**Restart Nginx Server**

|  |
| --- |
| $ pm2 start server.js |

Now, after pointing your domain to the APP\_SERVER\_PUBLIC\_IP and configure the server bloc to proxy the Node.js application, you will be able to see ‘Hello World’ on the browser by typing in your domain instead of [http://APP\_SERVER\_PUBLIC\_IP:8080](http://app_server_public_ip:8080).