Deploy a node app to AWS EC2, set up nginx server and configure https using letsencrypt certbot

sambuddhaadhikari.com/posts/deploy-node-app-ec2-nginx-https

Introduction

This article is a step-by-step guide to deploy a NodeJS app to AWS EC2 we will use the following tools or technologies:

- NodeJS + Express for the application
- AWS EC2 amazon-linux instance as the server
- nginx to set up reverse proxy
- letsencrypt certbot for security certificate

I am using macos, so other system users may need to make some changes as needed.

Prerequisites

- A github repository with a working nodejs application.
- Basic knowledge of command line, git and github.
- A domain name that you own (this domain should replace any instance of your.domain.com in this article)

If you do not have one, checkout my previous article.

Or directly fork my git repository.

Set up EC2 Instance

- Go to AWS and login or create a new account
- On the top left, select Services > Compute > EC2
- Click on "Launch Instance" and select "Launch Instance"
- Select a name of your choice, I am choosing `simple-node-app-server`
- Under Application and OS, Select Amazon Linux and keep the other options default
- Under Instance Type, select t2.micro
- Under Key pair (login), select Create new key pair
    - Enter any name of your choice, I am choosing `simple-node-server`
    - Choose Key pair type RSA
    - Choose Private key file format .pem, click create and then save it on your system somewhere safe. This is a secret file and someone can access your server if they have access to this.
    - Note the path to the file you can find this by right clicking on the .pem file and clicking info. This will be something like

```
/Users/your_user_name/path/to/folder/simple-node-server.pem
```

- Under network settings > Firewall, select Create security group and check all three options around allow ssh traffic, allow http traffic and allow https traffic. You might see an warning but this is fine, we will change this later.

Now click launch instance to start the EC2 instance, you should see a success message and the link to your new instance. Click on that to land on your instance details page.

- On the instances page, click on the instance id you just created
- On top right, click on connect and select SSH CLient, follow the instructions to connect to your instance via SSH
- Run `sudo yum update -y` to update everything

Now the instance set up is done and we are able to interact with the instance using SSH.

Clone repo from github

Set up SSH Keys and connect to github

SSH into your instance, then:

- Run `ssh-keygen -t ed25519`, in subsequent prompts select the default location and enter a passphrase twice. This passphrase is needed so remember it.
- Run `sudo nano ~/.ssh/id_ed25519.pub` and copy the contents of the public key, exit nano without saving by pressing `ctrl` and `x` together.
- Go to github > Settings > SSH and GPG Keys (URL)
- Click New SSH Key, select a title of your choice, select key type "Authentication Key" and add the contents of the public key in the text box then press Add SSH Key.

Now we are ready to connect to github using SSH.

Install node, npm and git

Run the following commands

```
#Install nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.2/install.sh | bash

#Enable nvm command
source ~/.bashrc

#Install node v16
nvm install 16

#Install git
sudo yum install git -y
```

Clone your github repository by running

```
# replace the part after "git clone" with your actual github repo endpoint.
git clone git@github.com:yourusername/simple-node-app.git
```

Set up the security group of the instance

- Go to AWS EC2 Console then click on instances and then your instance id to land on the instance summary page
- Scroll down and select the security tab
- Under security groups click on the link available
- Scroll down and click edit inbound rules
- It should already have HTTP, HTTPS and SSH configured. If not add them and select source anywhere
- Add a new rule of type custom TCP select port range 8080 or whatever port your app uses
- Click on save rules to save the same
- Go to your instance summary page again and copy the public IPV4 address. We will need this in the next step to replace `your.public.ip.here`

Start the app

SSH into your instance

- cd into the app by running `cd simple-node-app`
- Run `npm i` to install packages
- Start your app by running `node index.js` or whatever is your app's start script.

Now go to a browser and go to `http://your.public.ip.here:8080` and you should land on your app.

You can also choose to close out ssh applications to your server from any other ip but yours. You can do this by changing the source option of SSH from `0.0.0.0/0` to your ip. To find your ip go back to the ssh shell where you are connected to the instance and run `who am i`

- Go back to SSH shell and quit the server using `ctrl` + `c`
- install pm2 by running `npm install pm2 -g`
- Run your app again by running `pm2 start index.js`. Replace index.js with your app's endpoint.

Now the app will run even when you exit the shell command.

Set up nginx and certbot

We do not want to pass the PORT with the address when we deply. We would rather expose our 8080 port to the root IP address. To do this, we will use nginx.

SSH into your instance, then run the following commands

```
#Install nginx
sudo amazon-linux-extras install nginx1 -y

#Enable nginx
sudo systemctl enable nginx

#Start nginx
sudo systemctl start nginx

#Check nginx status
sudo systemctl status nginx
```

Now, go to your AWS Console EC2 instance and copy the public IP address and on any browser go to `http://your.public.ip.here` .

We should see the welcome page of nginx.

Note, above you must use http and not https as we have not set up ssl yet.

Edit nginx config

Now we will set up our nginx config to redirect http requests to https and allow https connections

- Run `sudo nano /etc/nginx/nginx.conf` to start editing the config file
- Remove everything from the file and put the following:

Replace your.domain.com with your actual domain name and under location, instead of 8080 use the PORT that your application uses

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    include /etc/nginx/conf.d/*.conf;

    # Redirect http requests to https
    server {
        listen       80 default_server;
        server_name  _;
        return 301 https://$host$request_uri;
    }

    # Set up TLS server
    server {
        listen       443 ssl http2;
        listen       [::]:443 ssl http2;
        server_name  your.domain.com;

        ssl_certificate "/etc/letsencrypt/live/your.domain.com/fullchain.pem";
        ssl_certificate_key "/etc/letsencrypt/live/your.domain.com/privkey.pem";
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers ECDH+AESGCM:ECDH+AES256:ECDH+AES128:DH+3DES:!ADH:!AECDH:!MD5;

        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout  10m;
        ssl_prefer_server_ciphers on;


        location / {
            proxy_pass http://localhost:8080;
        }
    }

}
```

Add DNS A Record

Go to your domain service provider and set up an A record which points to your EC2 Instance's public ipv4 address

Set up certbot

Certbot is a free, open-source software tool for automatically using Let's Encrypt certificates on manually-administrated websites to enable HTTPS.

- SSH into your instance
- If you are already there, make sure you are in the root directory, if not - `cd ~`

Run the following commands

```
#Download EPEL
sudo wget -r --no-parent -A 'epel-release-*.rpm'
https://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/

#Install the repository packages
sudo rpm -Uvh dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/epel-release-*.rpm

#Enable EPEL
sudo yum-config-manager --enable epel*

#Install certbot
sudo yum install -y certbot

#Install certbot-nginx
sudo yum install -y python-certbot-nginx
```

Now we have installed the certbot client and we will use it to create a certificate.

In the next command, make sure to replace your.domain.com with your actual domain name that youa re going to use

> Run `sudo certbot certonly --standalone --debug -d your.domain.com`

Now we have a TLS certificate.

> Run `sudo systemctl restart nginx` to effect the changes.

Now if you go to `your.domain.com` you should see your app deployed to EC2.

Set up cronjob to auto renew certificate

We will set up a cron job to try to auto renew the certificate every day at 12 noon.

- SSH into your instance
- Run `sudo crontab -e`
- To start editing, press esc and then press i
- paste the following into the editor `0 12 * * * /usr/bin/certbot renew --quiet`
- Press esc
- type `:wq` and press enter

Now we are all set.