Kimserey's blog

Software development, design and stuff

ASP NET Core with Nginx

June 01, 2018

ASP NET Core with Nginx

Few weeks ago I showed how to host ASP NET Core on Windows Server behind IIS. Compared to Windows Server, Ubuntu with nginx offers a quicker way to get started and a better control over the kestrel process.
Today we will see how to host an ASP NET Core application on Ubuntu. This post will be composed of three parts:

1. Install nginx
2. Configure nginx
3. Host ASP NET Core

1. Install nginx

Start by installing nginx.

```
 sudo apt-get update
sudo apt-get install nginx
```

After installing nginx, the daemon should have been installed and started. We should be able to navigate to `http://localhost` and see the nginx default page.
This page is the default root folder of nginx which can be found under `/var/www/html/` .

We should also be able to interact with it just like any other daemon managed by `systemd` :

```
 sudo systemctl start nginx
sudo systemctl stop nginx
sudo systemctl restart nginx
sudo systemctl status nginx
```

And similarly it can be debugged via `journald` :

```
sudo journalctl -xeu nginx
```

If you aren't familiar with systemd, you can read my previous blog post on how to Manage Kestrel process with systemd.

2. Configure nginx reverse proxy

Nginx is configured using configuration files known as `sites` .

It is made of modules containing directives allowing us to configure the behaviors of the proxy. The full documentation list all modules.

Here we will use `server` and `location` directives coming from the main module and some other directives coming the proxy module.

Nginx comes with a default configuration which can be found under the sites available `/etc/nginx/sites-available/default` . This is the default created with runs the index page from nginx.

We can create multiple configutations under the `sites-available` folder like so:

```
 /sites-available/default
/sites-available/myapp.com
/sites-available/helloworld.net
```

Those configurations will not yet be used by nginx. We need to enable them by having them in the `sites-enable` folder. This is achieved by creating a symlink, if we navigate to `/etc/nginx/sites-enabled` , we can find the symlink of `default` .

```
ln -s /etc/nginx/sites-available/myapp /etc/nginx/sites-enabled/m
```

In this example we will setup a frontend with an api. The frontend is a static html page which calls the backend via an api which is built in ASP NET Core.
To setup this, we can start by making our static page available by putting it into our app folder.
`/usr/share/myapp/www/index.html` .

Then we can setup the first location.

```
server {
```

```
    listen 80 default;

    listen [::]:80;

    server_name myapp.com www.myapp.com;

    root /usr/share/myapp/www;


    location / { }
}
```

Here we created a server which serves the default index on `myapp.com`. We listen on port 80 where `[::]:80` is meant for ipv6.
`root` is used to change the root folder to find the index file.
`server_name` defines the hostname which is handled by this configuration.
The location targets `/` with an empty content. This will then look for the default page which is the index file.

Now that the static files are served, we can proxy calls going to `/api` to our kestrel process:

```
server {
    listen 80;

    listen [::]:80;

    server_name myapp.com www.myapp.com;

    include /etc/nginx/conf.d/http;


    location / { }


    location /api/ {
        include /etc/nginx/proxy_params;

        proxy_pass http://localhost:5000/;
    }
}
```

`include` is a directive from the core module which allows us to include a set of directives.
We included `proxy_params` which is a set of directives condensed for proxying. This file can be found in /etc/nginx/proxy_params.

We also included a set of http params which set the connection to keep-alive.

```
 # /etc/nginx/conf.d/http
proxy_http_version 1.1;
proxy_set_header Connection keep-alive;
proxy_set_header Upgrade $http_upgrade;
proxy_cache_bypass $http_upgrade;
```

For `http://localhost:5000/`, the URI `/` is important as it indicates to nginx that we want to append the rest of the path and remove the matching part. Because of that, on `/api/`, the trailing slash is equally important as nginx appends a slash (else the proxy would be `http://localhost/api/a/b/c => http://localhost:5000//a/b/c` which won't be valid).

Once we are done we can check if the configuration is valid `nginx -t`. Next we can reload nginx with `sudo systemctl restart nginx`.
Provided that the symlink was already created, we should now be able to navigate to our static index and to our api endpoints.

3. Host ASP NET Core

Just to complete the loop, we can create a simple ASP NET Core application with a single endpoint returning a json object.

```
 public class HomeController: Controller
{
    [HttpGet("/home")]
    public IActionResult Get()
    {
        return Json(new { test = "test" });
    }
}
```

We can then uploaded on our server via scp. If you are not familiar with ssh on linux, you can refer to my previous blog post on ssh

```
 scp -r /myapp hostname1:~/
sudo mv ~/myapp /usr/share/myapp
```

We upload our binaries to `/usr/share/myapp` as it is the recommended linux filesystem path.

https://en.m.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

Then we can start the process using systemd like how I explained on my previous blog post on How to manage Kestrel with systemd.

And that conclude this post. Now that we have our app setup and nginx setup, we should be able to navigate to the site!

Conclusion

Today we saw how we could configure nginx to proxy our static content and our api. We saw how to work with nginx modules and directives and more importantly where to find the documentation. Lastly we saw how we could put our application on the server and have it proxied behind nginx. Hope you like this post, see you next time!

.NET Core .NET Standard C# Linux nginx Ubuntu

Enter your comment...

Popular posts from this blog

Manage assets and static files with Angular CLI

September 08, 2017

Manage assets and static files with Angular CLIOne of the easiest way to build Angular applicationns is through Angular CLI. Using the ng serve command will build and serve the whole application or we can use ng build to output the app into the outputDir ...

KEEP READING

Prime NG data table for Angular

December 01, 2017

Prime NG data table for AngularIn every web application there is a need to display data in a tabular form. If the amount of data is small, a simple combination of HTML with Bootstrap is enough. But when ...

KEEP READING

How to setup Continuous Integration/Deployment with GitLab for ASP NET Core application

December 07, 2017

How to setup Continuous Integration/Deployment with GitLab for ASP NET Core applicationIn any application, Continuous Integration and Continous Deployment environments are important to setu ...

KEEP READING

Powered by Blogger

Theme images by Michael Elkan

KIMSEREY LAM

Follow 26

VISIT PROFILE

Labels

Archive

Kimserey Lam

Senior Software Engineer at Jet.com

Jet.com

ESIEE PARIS

View profile

Follow @kimserey_lam

BUY ME A COFFEE

KIMSEREY
KIMSEREYLAM.COM

Report Abuse