How to Deploy a .NET Core Web Application on CentOS 7

Last Updated: Fri, May 11, 2018

CentOS Linux Guides Server Apps

.NET Core is a redesigned open source cross-platform development framework maintained by Microsoft and the ever-growing .NET community. With the help of .NET Core, developers can easily build modern high-performance applications on all kinds of platforms.

In this article, I will show you how to install .NET Core on a CentOS 7 server instance and then deploy a full functional .NET Core web application.

Prerequisites

- A fresh Vultr CentOS 7 x64 server instance. Say its IP address is `203.0.113.1`.
- A sudo user.
- The server instance has been updated to the latest stable status using the EPEL YUM repo.

Add the .NET product feed to the system

As a cross-platform development framework, .NET Core provides pre-compiled binaries for various operating systems. On CentOS 7, you can setup an officially signed .NET YUM repo by running the following commands as a sudo user:

```
sudo rpm --import https://packages.microsoft.com/ke
sudo sh -c 'echo -e "[packages-microsoft-com-prod]\
```

Install the latest .NET SDK using YUM

Having the .NET YUM repo in place, install the latest .NET SDK, including .NET Core and other dependencies, on your machine:

```
sudo yum update -y
sudo yum install libunwind libicu -y
sudo yum install dotnet-sdk-2.1.4 -y
```

In order to confirm the result, you can create and run a "Hello World" demo .NET Core app:

```
cd
dotnet new console -o helloworldApp
cd helloworldApp
dotnet run
```

The `dotnet new console -o helloworldApp` command will create a directory named `helloworldApp` in your home directory and then use the `console` template to generate app files in the newly created directory.

Upon executing the `dotnet run` command, you will see the `Hello World!` message in the console.

Create a .NET Core web app

Now, create and run a .NET Core application of type `razor`. Just remember that "Razor Pages" is a new application template of .NET Core MVC that is designed for page-oriented scenarios:

```
cd
dotnet new razor -o myfirstwebapp
cd myfirstwebapp
dotnet run
```

By executing the `dotnet run` command above, you will start a .NET Core web app listening on: `http://localhost:5000`.

If you want to confirm that the web app is up and running, although it's on a server instance with no GUI, you can still open a new terminal console and input `curl http://localhost:5000` to view the source code of the web app's home page.

Afterwards, you can press `CTRL` + `C` to shut down the .NET Core web app.

To materialize your web app, you need to edit files within the app directory. You can learn more details in the official .NET document page.

Having all of the development tasks done, you can use the following commands to publish your web app:

```
cd ~/myfirstwebapp
dotnet publish
```

You can find the published web app in the `~/myfirstwebapp/bin/Debug/netcoreapp2.0` directory.

(Optional): Setup Supervisor to keep your .NET Core web app online

Process crashes happen. In order to keep your web app online, it's a good idea to have a process managment tool, such as Supervisor, to monitor and restart the crashed web app processes.

On CentOS 7, you can install Supervisor using YUM:

```
sudo yum install supervisor -y
```

Next, you need to setup a dedicated Supervisor config file for your web app:

```
cd /etc/supervisord.d
sudo vi myfirstwebapp.conf
```

Populate the file:

```
[program:myfirstwebapp]
```

```
command=dotnet myfirstwebapp.dll
directory=/home/sudouser/myfirstwebapp/bin/Debug/ne
environment=ASPNETCORE__ENVIRONMENT=Production
user=root
stopsignal=INT
autostart=true
autorestart=true
startsecs=1
stderr_logfile=/var/log/myfirstwebapp.err.log
stdout_logfile=/var/log/myfirstwebapp.out.log
```

Save and quit:

```
:wq!
```

Next, you need to modify the default `supervisord` config file to include the config file we've created:

```
sudo cp /etc/supervisord.conf /etc/supervisord.conf
sudo vi /etc/supervisord.conf
```

Find the last line:

```
files = supervisord.d/*.ini
```

Replace it:

```
files = supervisord.d/*.conf
```

Save and quit:

```
:wq!
```

Start Supervisor and set it to automatically start at system startup:

```
sudo systemctl start supervisord.service
sudo systemctl enable supervisord.service
```

Load the new Supervisor settings:

```
sudo supervisorctl reread
sudo supervisorctl update
```

Now, you can use the following command to show the app's status:

```
sudo supervisorctl status
```

The output will look like the following:

```
myfirstwebapp                    RUNNING   pid 3925
```

Next, you can try to kill the app's process by specifying the pid `3925`:

```
sudo kill -s 9 3925
```

Wait for a while, and then check the status again:

```
sudo supervisorctl status
```

This time, the output will indicate that the app did break down and automatically started:

```
myfirstwebapp                    RUNNING   pid 392
```

(Optional): Install Nginx as a reverse proxy

In order to facilitate visitors' access, you can install Nginx as a reverse proxy to pass web traffic to port `5000`.

Install Nginx using YUM:

```
sudo yum install nginx -y
```

Edit the default Nginx config file as follows:

```
sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.con
sudo vi /etc/nginx/nginx.conf
```

Find the following segment within the `http {}` segment:

```
location / {
}
```

Insert six lines of reverse proxy settings between the braces as shown below:

```
location / {
proxy_pass http://127.0.0.1:5000;
```

```
proxy_redirect off;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwa
proxy_set_header X-Forwarded-Proto $scheme;
}
```

Save and quit:

```
:wq!
```

Start the Nginx service and then set it to start at system startup:

```
sudo systemctl start nginx.service
sudo systemctl enable nginx.service
```

Setup firewall rules

Before visitors can access the .NET Core web app on ports `80` and `443`, you need to modify firewall rules as shown below:

```
sudo firewall-cmd --zone=public --permanent --add-s
sudo firewall-cmd --zone=public --permanent --add-s
sudo firewall-cmd --reload
```

This completes the application setup. You're now ready to browse your .NET Core web app at `http://203.0.113.1`.

Want to contribute?

You could earn up to **$600** by adding new articles

Submit your article

Suggest an update

Request an article

Products

Features

Marketplace

Resources

Company

Over 45,000,000 Cloud Servers Launched

Terms of Service

AUP/DMCA

Privacy Policy

Cookie Policy

© Vultr 2021

VULTR is a registered trademark of The Constant Company, LLC.