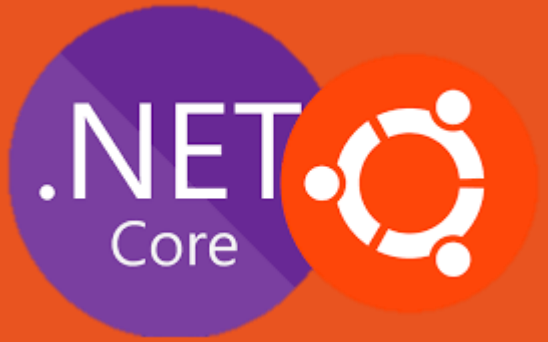
.NET Core development on Ubuntu 22.04

Yeongseon Choe
Microsoft Korea



Topics

1. Prerequisite
2. Introduction to .NET
3. Install the .NET 6 is on Ubuntu
4. C# Console Application

Prerequisite

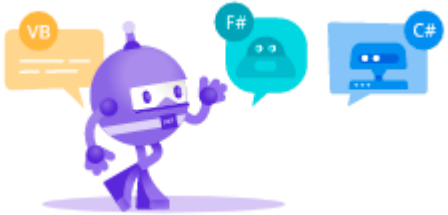
- Ubuntu 22.04 LTS
- Docker
- Visual Studio Code

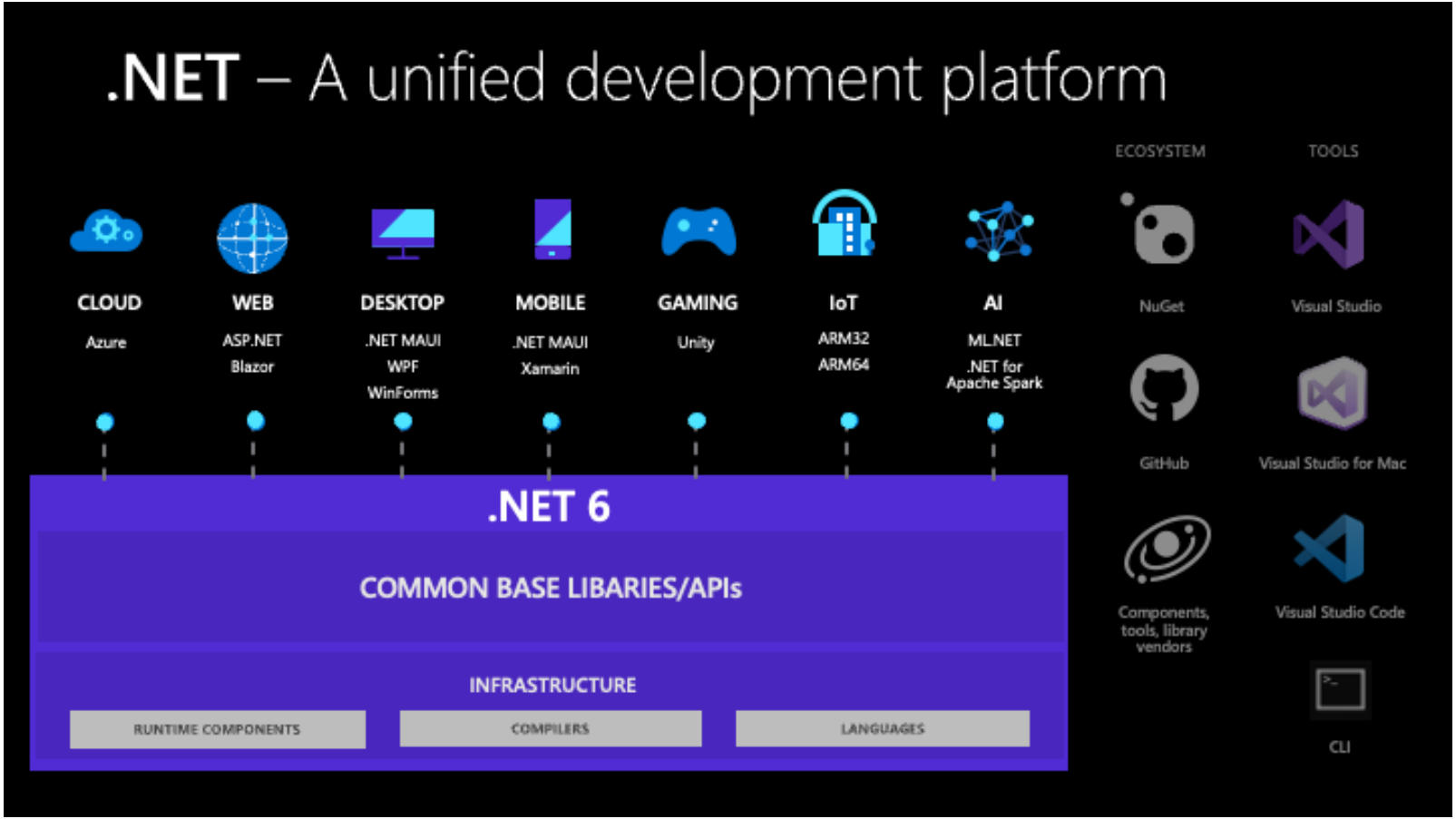
What is the .NET?

- In 2002, Microsoft released .NET Framework, a development platform for creating Windows apps. Today .NET Framework is at version 4.8 and remains fully supported by Microsoft.
- In 2014, Microsoft introduced .NET Core as a cross-platform, open-source successor to .NET Framework. This new implementation of .NET kept the name .NET Core through version 3.1. The next version after .NET Core 3.1 was named .NET 5.
- New .NET versions continue to be released annually, each a major version number higher. They include significant new features and often enable new scenarios.

What is .NET (Core) ?

- Cross-Platform
- OpenSource
- Multiple language (C#, F#, or Visual Basic)
- Building for web, mobile, desktop, games, IoT, and more.




.NET – A unified development platform
CLOUD
Azure
WEB
ASP.NET
Blazor
DESKTOP
.NET MAUI
WPF
WinForms
MOBILE
.NET MAUI
Xamarin
GAMING
Unity
IoT
ARM32
ARM64
AI
ML.NET
.NET for Apache Spark
.NET 6
COMMON BASE LIBARIES/APIs
INFRASTRUCTURE
RUNTIME COMPONENTS
COMPILERS
LANGUAGES
ECOSYSTEM
TOOLS
NuGet
Visual Studio
GitHub
Visual Studio for Mac
Components, tools, library vendors
Visual Studio Code
CLI

[Page title here]

Microsoft and Canonical announce native .NET availability in Ubuntu 22.04 hosts and containers

Canonical

on 16 August 2022

.NET

Ubuntu

[Page title here]

- .NET developers are now able to install the ASP.NET and .NET SDK and runtimes from Ubuntu 22.04 LTS with a single “apt install” command

- Canonical releases new, ultra-small OCI-compliant appliance images, without a shell or package manager, for both the .NET 6 LTS and ASP.NET runtimes

- Microsoft and Canonical are collaborating to secure the software supply chain between .NET and Ubuntu and to provide enterprise-grade support

Install the .NET 6 on Ubuntu 22.04

```
# quickly install a bundle with both the SDK and the runtime

sudo apt update && sudo apt install dotnet6


# or cherry-pick only the dependencies you need to develop or run

sudo apt install dotnet-sdk-6.0

sudo apt install dotnet-runtime-6.0

sudo apt install aspnetcore-runtime-6.0
```

Install the .NET 6 on Ubuntu 22.04

- Supported distributions

Ubuntu	.NET
22.04 (LTS)	6+
20.04 (LTS)	3.1, 6
18.04 (LTS)	3.1, 6
16.04 (LTS)	3.1, 6

Install the .NET 6 on Ubuntu (20.04, 18.04, 16.04)

```
// add the Microsoft package signing key to your list of trusted keys and add the package repository
wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft-prod.deb

// install the SDK
sudo apt-get update && sudo apt-get install –y dotnet-sdk-6.0

// install the runtime
sudo apt-get update && sudo apt-get install –y aspnetcore-runtime-6.0
```

Install .NET 6 on Ubuntu 22.04

```
yeongseon@vm-ubuntu:~$ dotnet
Command 'dotnet' not found, but can be installed with:
sudo snap install dotnet-sdk
yeongseon@vm-ubuntu:~$ sudo apt update && sudo apt install dotnet6
yeongseon@vm-ubuntu:~$ dotnet

Usage: dotnet [options]
Usage: dotnet [path-to-application]

Options:
  -h|--help         Display help.
  --info            Display .NET information.
  --list-sdks       Display the installed SDKs.
  --list-runtimes   Display the installed runtimes.

path-to-application:
  The path to an application .dll file to execute.
```

dotnet command

Command	Function
dotnet build	Builds a .NET application.
dotnet build-server	Interacts with servers started by a build.
dotnet clean	Clean build outputs.
dotnet exec	Runs a .NET application.
dotnet help	Shows more detailed documentation online for the command.
dotnet migrate	Migrates a valid Preview 2 project to a .NET Core SDK 1.0 project.
dotnet msbuild	Provides access to the MSBuild command line.
dotnet new	Initializes a C# or F# project for a given template.
dotnet pack	Creates a NuGet package of your code.
dotnet publish	Publishes a .NET framework-dependent or self-contained application.
dotnet restore	Restores the dependencies for a given application.
dotnet run	Runs the application from source.
dotnet sdk check	Shows up-to-date status of installed SDK and Runtime versions.
dotnet sln	Options to add, remove, and list projects in a solution file.
dotnet store	Stores assemblies in the runtime package store.
dotnet test	Runs tests using a test runner.

Install the .NET 6 on Ubuntu

```
yeongseon@vm-ubuntu:~$ dotnet new
The 'dotnet new' command creates a .NET project based on a template.

Common templates are:
Template Name        Short Name    Language    Tags
-------------------  ------------  ----------  -------------------
ASP.NET Core Web App  webapp,razor  [C#]       Web/MVC/Razor Pages
Blazor Server App     blazorserver  [C#]       Web/Blazor
Class Library         classlib      [C#],F#,VB  Common/Library
Console App           console       [C#],F#,VB  Common/Console

An example would be:
  dotnet new console

Display template options with:
  dotnet new console -h
Display all installed templates with:
  dotnet new --list
Display templates available on NuGet.org with:
  dotnet new web --search
```

Install the .NET 6 on Ubuntu

```
yeongseon@vm-ubuntu:~$ dotnet new console -o HelloUbuCon -f net6.0

yeongseon@vm-ubuntu:~$ cd HelloUbuCon
yeongseon@vm-ubuntu:~/ HelloUbuCon $ ls
HelloUbuCon.csproj  Program.cs  obj

yeongseon@vm-ubuntu:~/HelloUbuCon$ dotnet run
Hello, World!

yeongseon@vm-ubuntu:~/HelloUbuCon$ cat Program.cs
// See https://aka.ms/new-console-template for more information
Console.WriteLine("Hello, UbuCon Asia 2022!");

yeongseon@vm-ubuntu:~/HelloUbuCon$ dotnet run
Hello, UbuCon Asia 2022!
```

Install the .NET 6 on Ubuntu

```
yeongseon@vm-ubuntu:~/HelloUbuCon$ cat Program.cs
namespace HelloUbunCon
{
   class Program
   {
      static void Main(string[] args)
      {
         String message = "Hello, UbuCon Asis 2022!";
         Console.WriteLine(message);
      }
   }
}

yeongseon@vm-ubuntu:~/HelloUbuCon$ dotnet run
Hello, UbuCon Asis 2022!
```

C# Program

```
using System;
using System.Text;
```
Importing Namespace Section

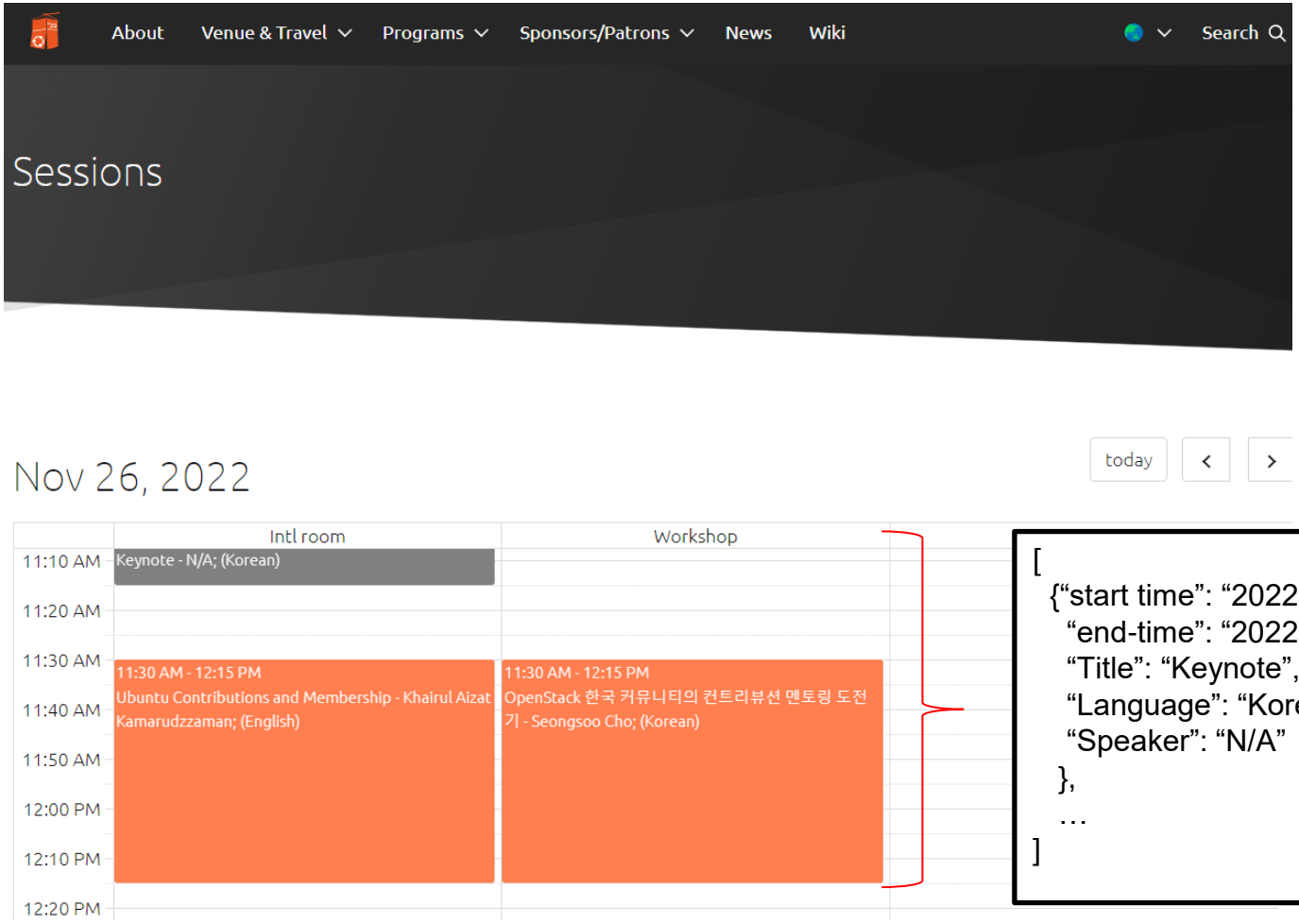
```
namespace CSharpProgram
{
```
Namespace Declaration Section

```
    internal class Program
    {
```
Class Declaration Secion

```
        static void Main(string[] args)
        {
            string message = "Hello World!!";
            Console.WriteLine(message);
        }
    }
}
```
Main Method Section

Create a C# console app

```
yeongseon@vm-ubuntu:~$ dotnet new console -o ConsoleApp -f net6.0
The template "Console App" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on /home/yeongseon/ConsoleApp/ConsoleApp.csproj...
  Determining projects to restore...
  Restored /home/yeongseon/ConsoleApp/ConsoleApp.csproj (in 124 ms).
Restore succeeded.


yeongseon@vm-ubuntu:~$ cd  ConsoleApp/
yeongseon@vm-ubuntu:~/ConsoleApp$ code .
```




```
[
 {“start time”: “2022-11-26 10:00:00”,
  “end-time”: “2022-11-26 10:30:00”,
  “Title”: “Keynote”,
  “Language”: “Korean”,
  “Speaker”: “N/A”
 },
 …
]
```

Implement Main

```
namespace ConsoleApp
{
   internal class Program
   {
      public class Session
      {
         public DateTime StartTime { get; set; }
         public DateTime EndTime { get; set; }
         public string? Title { get; set; }
         public string? Language { get; set; }
         public string? Speaker { get; set; }
      }
      static void Main(string[] args)
      {
      }
   }
}
```

Demo

What is the ASP.NET?

- ASP.NET is a free web framework for building great websites and web applications using HTML, CSS, and JavaScript.

- ASP.NET Core is an alternative to ASP.NET

ASP.NET Core

- Build web apps and services, Internet of Things (IoT) apps, and mobile backends.
- Use your favorite development tools on Windows, macOS, and Linux.
- Deploy to the cloud or on-premises.
- Run on .NET Core. (dotnet run)

ASP.NET Core vs ASP.NET 4.x

ASP.NET Core	ASP.NET 4.x
Build for Windows, macOS, or Linux	Build for Windows
Razor Pages is the recommended approach to create a Web UI as of ASP.NET Core 2.x. See also MVC, Web API, and SignalR.	Use Web Forms, SignalR, MVC, Web API, WebHooks, or Web Pages
Multiple versions per machine	One version per machine
Develop with Visual Studio, Visual Studio for Mac, or Visual Studio Code using C# or F#	Develop with Visual Studio using C#, VB, or F#
Higher performance than ASP.NET 4.x	Good performance
Use .NET Core runtime	Use .NET Framework runtime

Developing ASP.NET Core apps

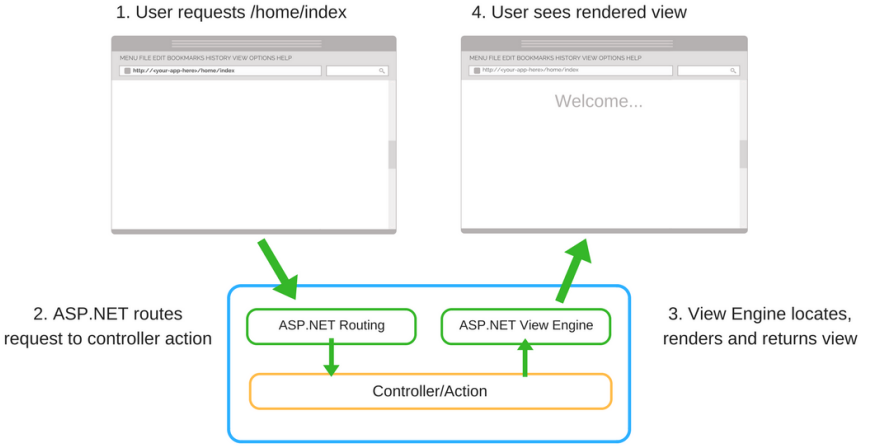
App type	Scenario	Tutorial
Web app	New server-side web UI development	Get started with Razor Pages
Web app	Maintaining an MVC app	Get started with MVC
Web app	Client-side web UI development	Get started with Blazor
Web API	RESTful HTTP services	Create a web API†
Remote Procedure Call app	Contract-first services using Protocol Buffers	Get started with a gRPC service
Real-time app	Bidirectional communication between servers and connected clients	Get started with SignalR

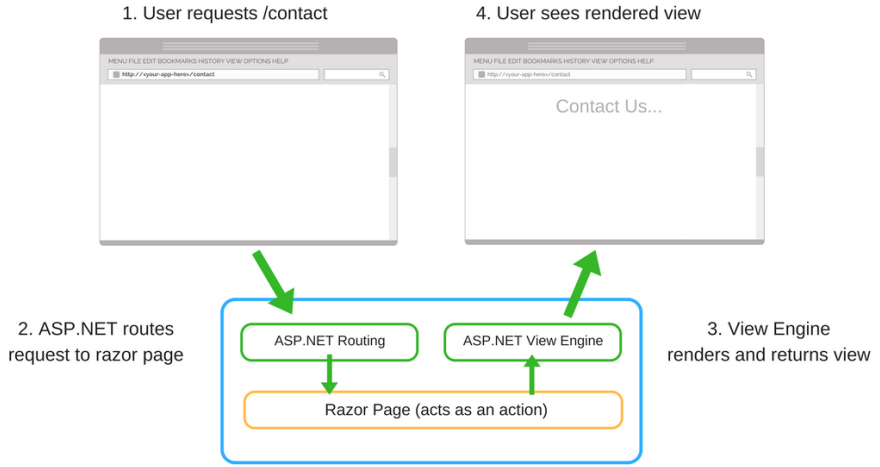
Razor Pages App

- Razor Pages provides a simpler way to organize code within ASP.NET Core applications.
- keeping implementation logic and view models closer to the view implementation code.

MVC vs. Rage Pages

MVC	Razor Pages
• /Controllers/CartController.cs	• /Pages/Cart/Index.cshtml
• /ViewModels/CartViewModel.cs	↳ Index.cshtml.cs
• /Views/Cart/Index.cshtml	
3 Root Level Folders	1 Root Level Folder

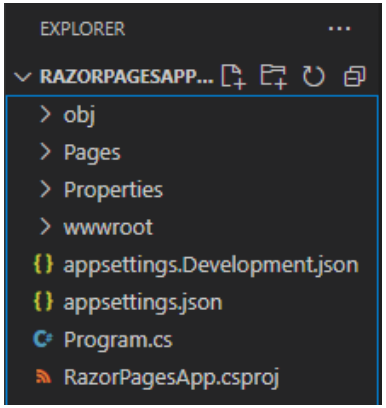




Create a Razor Pages App

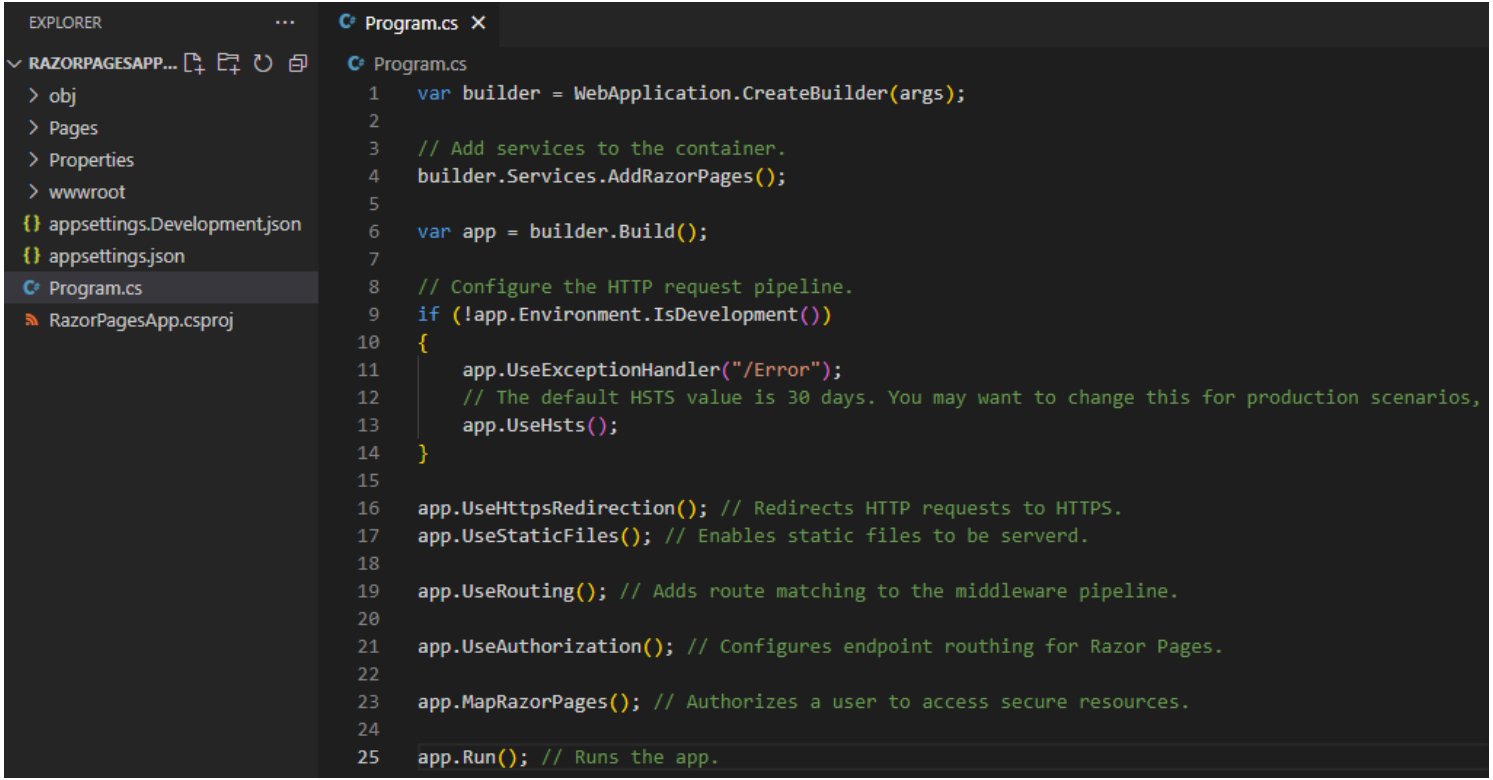
```
yeongseon@vm-ubuntu:~$ dotnet new webapp -o RazorPagesApp
The template "ASP.NET Core Web App" was created successfully.
This template contains technologies from parties other than Microsoft, see https://aka.ms/aspnetcore/6.0-third-party-notices for details.

yeongseon@vm-ubuntu:~/RazorPagesApp$ cd ..
yeongseon@vm-ubuntu:~$ cd RazorPagesApp/
yeongseon@vm-ubuntu:~/RazorPagesApp$ ls
Pages  Program.cs  Properties  RazorPagesApp.csproj  appsettings.Development.json  appsettings.json  obj  wwwroot
yeongseon@vm-ubuntu:~/RazorPagesApp$
```

Project files for Razor Pages

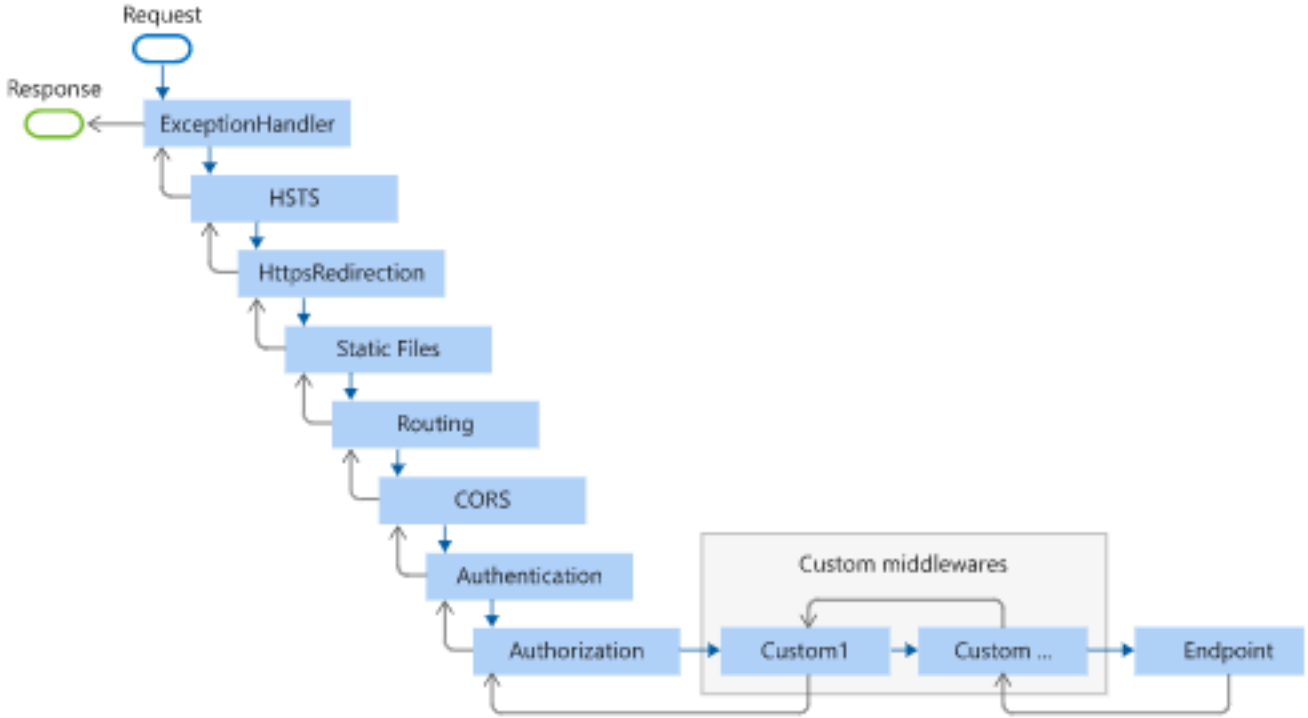


- Pages folder: Razor page is a pair of .cshtml, .cshtml.cs
  - .cshtml: HTML markup with C# code using Razor syntax
  - .cshtml.cs: C# code that handles page events.
- wwwroot folder: Static assets including HTML, JavaScript, and CSS.
- appsettings.json: Configuration data such as connection strings.
- Program.cs: Entry point of application

Program.cs


EXPLORER
RAZORPAGESAPP...
obj
Pages
Properties
wwwroot
appsettings.Development.json
appsettings.json
Program.cs
RazorPagesApp.csproj
Program.cs
Program.cs
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddRazorPages();
var app = builder.Build();
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
app.UseExceptionHandler("/Error");
// The default HSTS value is 30 days. You may want to change this for production scenarios,
app.UseHsts();
}
app.UseHttpsRedirection(); // Redirects HTTP requests to HTTPS.
app.UseStaticFiles(); // Enables static files to be serverd.
app.UseRouting(); // Adds route matching to the middleware pipeline.
app.UseAuthorization(); // Configures endpoint routhing for Razor Pages.
app.MapRazorPages(); // Authorizes a user to access secure resources.
app.Run(); // Runs the app.

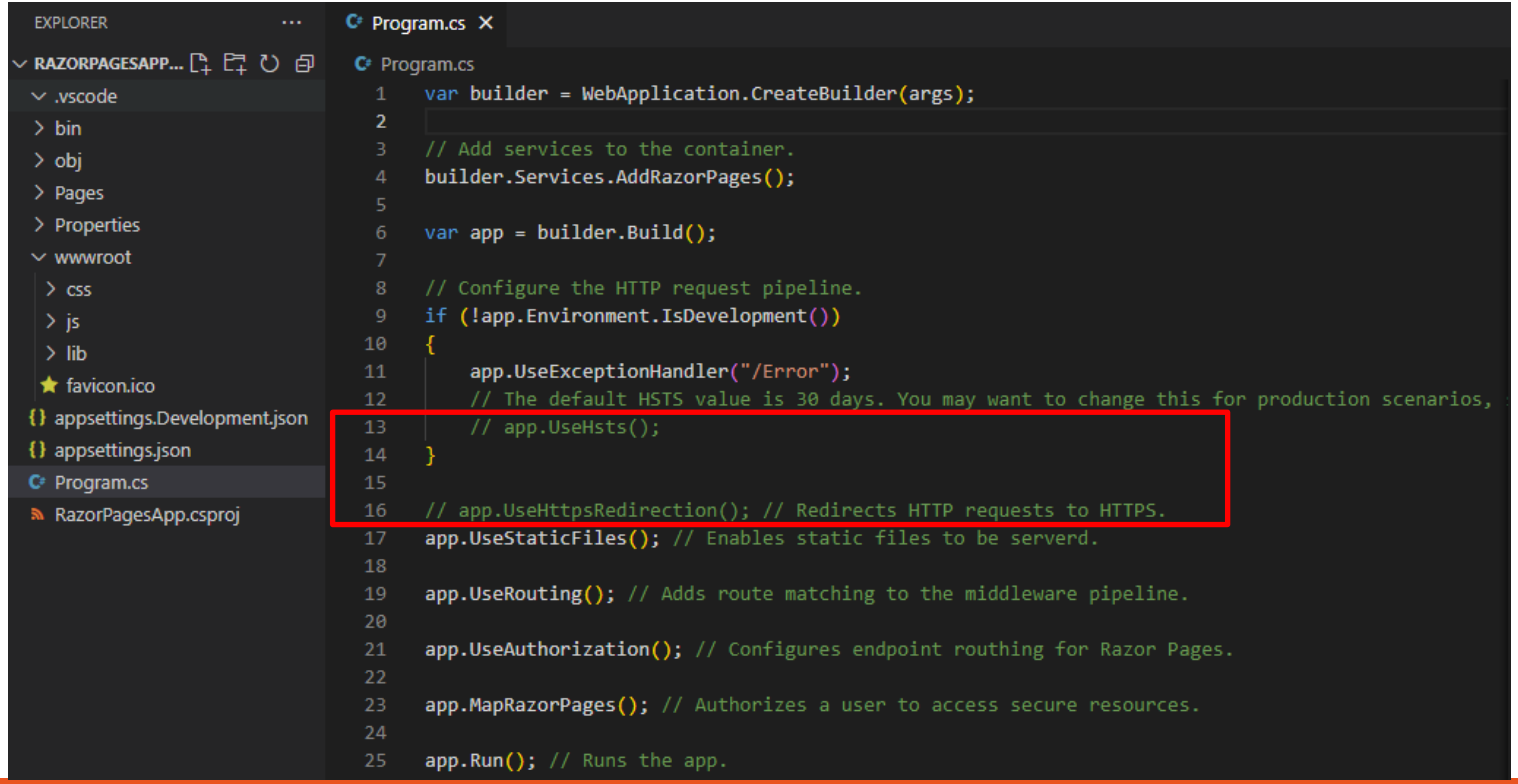
Middleware in ASP.NET Core apps



Run a Razor Pages Web on Ubuntu

```
yeongseon@vm-ubuntu:~/RazorPagesApp$dotnet run

yeongseon@vm-ubuntu:~/RazorPagesApp$ netstat -tlp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      -
tcp        0      0 localhost:36659         0.0.0.0:*               LISTEN      1189/node
tcp        0      0 localhost:7242          0.0.0.0:*               LISTEN      3890/RazorPagesApp
tcp        0      0 localhost:5217          0.0.0.0:*               LISTEN      3890/RazorPagesApp
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN      -
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN      -
tcp6       0      0 ip6-localhost:7242      [::]:*                  LISTEN      3890/RazorPagesApp
tcp6       0      0 ip6-localhost:5217      [::]:*                  LISTEN      3890/RazorPagesApp
```

Run a Razor Pages Web on Ubuntu

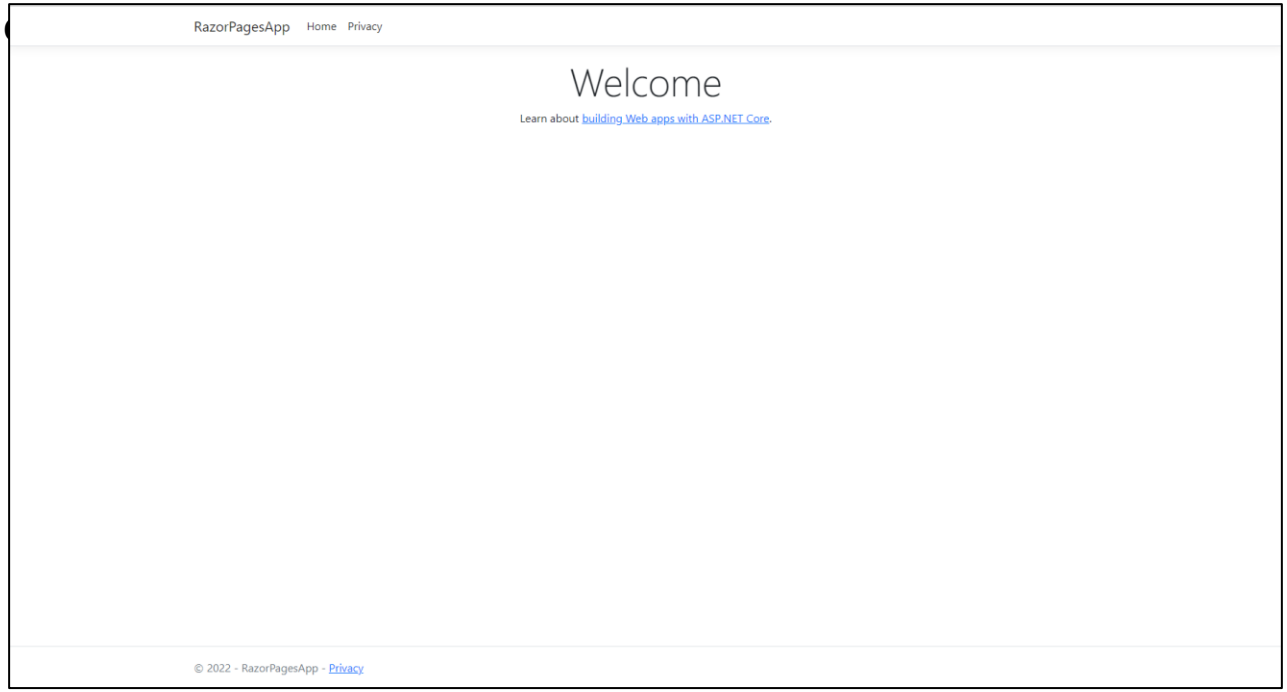


Run a Razor Pages Web on Ubuntu

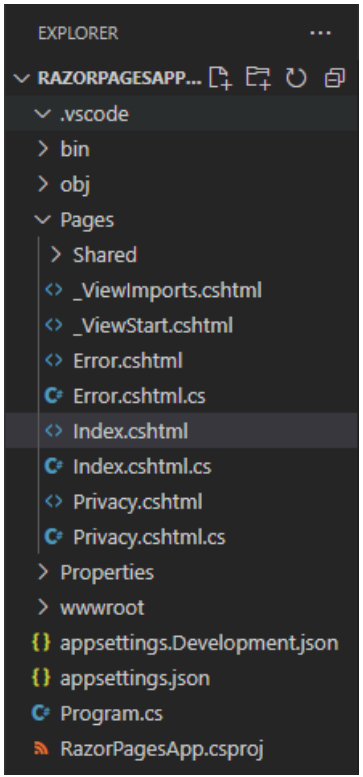
```
yeongseon@vm-ubuntu:~/RazorPagesApp$dotnet watch run
```

Run a Razor Pages Web on Ubuntu

[Content h



cshtml, cshtml.cs Razor Pages App



- Index.cshtml : Pages
- Index.cshtml.cs : Pages models

Add a model to a Razor Pages app

```
// Add a folder named Models.
yeongseon@vm-ubuntu:~/RazorPagesApp$ mkdir Models

yeongseon@vm-ubuntu:~/RazorPagesApp$ dotnet new --install Yae.Templates
The following template packages will be installed:
   Yae.Templates
Success: Yae.Templates::0.0.2 installed the following templates:
Template Name      Short Name  Language  Tags
-----------------  ----------  --------  -----------
Class.cs file      class       [C#]      Common/Code
Enum.cs file       enum        [C#]      Common/Code
Interface.cs file  interface   [C#]      Common/Code
Struct.cs file     struct      [C#]      Common/Code

yeongseon@vm-ubuntu:~/RazorPagesApp$ dotnet new class -t Session -o Models
The template "Class.cs file" was created successfully.
```

Add a model to a Razor Pages app

```
yeongseon@vm-ubuntu:~/RazorPagesApp$ cat Models/Session.cs
namespace RazorPagesApp.Models
{
    public class Session
    {
        public DateTime StartTime { get; set; }
        public DateTime EndTime {get; set; }
        public string? Title { get; set; }
        public string? Language { get; set;}
        public string? Speaker {get; set;}

        public override string ToString()
        {
            JsonSerializer.Serialize<Session>(this);
        }
    }
}
```

Demo

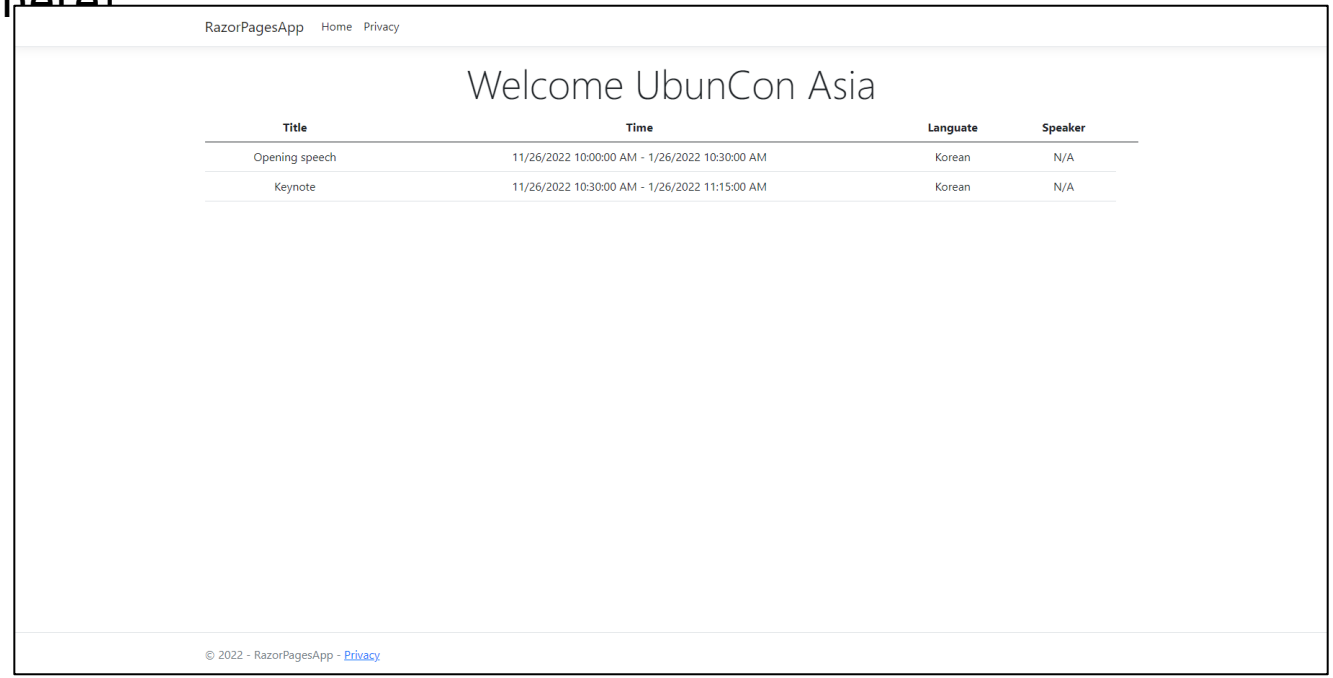
[Content here]

Add a model to a Razor Pages app

```
yeongseon@vm-ubuntu:~/RazorPagesApp$dotnet watch run
```

Adding a model to a Razor Pages App

[Content here]



Host and deploy ASP.NET Core

- Deploy ASP.NET Core app to VM
- Deploy ASP.NET Core app to Docker Container
- Deploy ASP.NET Core app to Azure App Service

Deploy ASP.NET Core app to VM

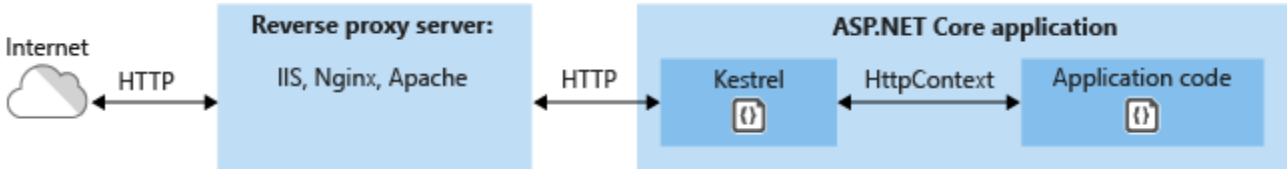
```
// Create publishing files in the ~/RazorPagesApp/bin/ Release /net6.0/publish/ folder
yeongseon@vm-ubuntu:~/RazorPagesApp$ dotnet publish -c Release
Microsoft (R) Build Engine version 17.0.1+b177f8fa7 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  All projects are up-to-date for restore.
  RazorPagesApp -> /home/yeongseon/RazorPagesApp/bin/Release/net6.0/RazorPagesApp.dll
  RazorPagesApp -> /home/yeongseon/RazorPagesApp/bin/Release/net6.0/publish/

// Copy all files to /var/RazorPagesApp/ folder
yeongseon@vm-ubuntu:~/RazorPagesApp$ sudo cp -a bin/ Release /net6.0/publish/ /var/RazorPagesApp/

// Run the app from a published folder
yeongseon@vm-ubuntu:~/RazorPagesApp$ dotnet /var/RazorPagesApp/RazorPagesApp.dll
```

Deploy ASP.NET Core app to VM

```
// Installi the nginx
yeongseon@vm-ubuntu:~$ sudo apt install nginx

yeongseon@vm-ubuntu:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2022-11-13 12:04:39 UTC; 7min ago
       Docs: man:nginx(8)
    Process: 1694 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, st>
    Process: 1695 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SU>
   Main PID: 1791 (nginx)

// Testing the nginx installation
yeongseon@vm-ubuntu:~$ curl localhost
<!DOCTYPE html>
```

Deploy ASP.NET Core app to VM

- Configuring Nginx as reverse proxy to route the request that made on port 80 to ASP.NET Core app that is listening on port 5000.



- Reverse proxy server can offload work such as serving static content, caching requests, compressing requests, and HTTPS termination from the HTTP server

Deploy ASP.NET Core app to VM

```
// Configure Nginx as reverse proxy to route the requests to ASP.NET Core app
// Nginx configuration file
yeongseon@vm-ubuntu:~/RazorPagesApp$ cat /etc/nginx/nginx.conf
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;
…
        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
```

Deploy ASP.NET Core app to VM

```
// Edit nginx configuration
yeongseon@vm-ubuntu:~$ sudo vi /etc/nginx/sites-enabled/default
server {
   listen       80;
   server_name _;
   location / {
      proxy_pass       http://localhost:5000;
      proxy_http_version 1.1;
      proxy_set_header   Upgrade $http_upgrade;
      proxy_set_header   Connection keep-alive;
      proxy_set_header   Host $host;
      proxy_cache_bypass $http_upgrade;
      proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header   X-Forwarded-Proto $scheme;
   }
}
```

Deploy ASP.NET Core app to VM

```
// Test Nginx configuration file
yeongseon@vm-ubuntu:~$ sudo nginx –t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

// Restart Nginx
yeongseon@vm-ubuntu:~$ sudo systemctl restart nginx
```

Deploy ASP.NET Core app to VM

```
yeongseon@vm-ubuntu:~$ curl localhost
<html>
<head><title>502 Bad Gateway</title></head>
<body>
<center><h1>502 Bad Gateway</h1></center>
<hr><center>nginx/1.18.0 (Ubuntu)</center>
</body>
</html>
yeongseon@vm-ubuntu:~$ wget localhost
--2022-11-14 01:17:51--  http://localhost/
Resolving localhost (localhost)... 127.0.0.1
Connecting to localhost (localhost)|127.0.0.1|:80... connected.
HTTP request sent, awaiting response... 502 Bad Gateway
2022-11-14 01:17:51 ERROR 502: Bad Gateway.
```

Deploy ASP.NET Core app to VM

```
// Check the Nginx logs
yeongseon@vm-ubuntu:~$ cat /var/log/nginx/access.log

yeongseon@vm-ubuntu:~$ cat /var/log/nginx/error.log
2022/11/13 12:04:39 [notice] 1791#1791: using inherited sockets from "6;7;"
2022/11/14 01:15:31 [error] 2614#2614: *1 connect() failed (111: Unknown error) while connecting to upstream,
client: 127.0.0.1, server: _, request: "GET / HTTP/1.1", upstream: "http://127.0.0.1:5000/", host: "localhost"
2022/11/14 01:17:51 [error] 2614#2614: *3 connect() failed (111: Unknown error) while connecting to upstream,
client: 127.0.0.1, server: _, request: "GET / HTTP/1.1", upstream: "http://127.0.0.1:5000/", host: "localhost"
```

Deploy ASP.NET Core app to VM

```
// Workaround: Restarting ASP.NET Core app manually
yeongseon@vm-ubuntu:~$ dotnet /var/RazorPagesApp/RazorPagesApp.dll

// Inspecting
yeongseon@vm-ubuntu:~/RazorPagesApp$ curl localhos
```

Deploying the app to VM

```
yeongseon@vm-ubuntu:~$ cat /lib/systemd/system/nginx.service
[Unit]
Description=A high performance web server and a reverse proxy server
Documentation=man:nginx(8)
After=network.target nss-lookup.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t -q -g 'daemon on; master_process on;'
ExecStart=/usr/sbin/nginx -g 'daemon on; master_process on;'
ExecReload=/usr/sbin/nginx -g 'daemon on; master_process on;' -s reload
ExecStop=-/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid
TimeoutStopSec=5
KillMode=mixed

[Install]
WantedBy=multi-user.target
```

Deploying the app to VM

```
yeongseon@vm-ubuntu:~$ cat vi /etc/systemd/system/razorpagesapp.service
[Unit]
Description=RazorPagesApp running on Ubuntu

[Service]
WorkingDirectory=/var/RazorPagesApp/
ExecStart=/usr/bin/dotnet /var/RazorPagesApp/RazorPagesApp.dll
Restart=always
# Restart service after 10 seconds if the dotnet service crashes:
RestartSec=10
KillSignal=SIGINT
SyslogIdentifier=razorpagesapp-identifier
User=www-data
Environment=ASPNETCORE_ENVIRONMENT=Development
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

[Install]
WantedBy=multi-user.target
```

Deploying the app to VM

```
yeongseon@vm-ubuntu:~$ systemctl status razorpagesapp.service
○ razorpagesapp.service - RazorPagesApp running on Ubuntu
     Loaded: loaded (/etc/systemd/system/razorpagesapp.service; disabled; vendor preset: enabled)
     Active: inactive (dead)

yeongseon@vm-ubuntu:~$ sudo systemctl start razorpagesapp.service

yeongseon@vm-ubuntu:~$ systemctl status razorpagesapp.service
● razorpagesapp.service - RazorPagesApp running on Ubuntu
     Loaded: loaded (/etc/systemd/system/razorpagesapp.service; disabled; vendor preset: enabled)
     Active: active (running) since Mon 2022-11-14 03:47:39 UTC; 2s ago
   Main PID: 5543 (dotnet)
      Tasks: 17 (limit: 4095)
     Memory: 21.1M
        CPU: 439ms
     CGroup: /system.slice/razorpagesapp.service
             └─5543 /usr/bin/dotnet /var/RazorPagesApp/RazorPagesApp.dll
```

Deploying the app to Docker Container

```
// Create publishing files in the ~/RazorPagesApp/bin/ Release /net6.0/publish/ folder
yeongseon@vm-ubuntu:~/RazorPagesApp$ dotnet publish -c Release
Microsoft (R) Build Engine version 17.0.1+b177f8fa7 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  All projects are up-to-date for restore.
  RazorPagesApp -> /home/yeongseon/RazorPagesApp/bin/Release/net6.0/RazorPagesApp.dll
  RazorPagesApp -> /home/yeongseon/RazorPagesApp/bin/Release/net6.0/publish/

yeongseon@vm-ubuntu:~/RazorPagesApp$ ls bin/Release/net6.0/publish/
RazorPagesApp         RazorPagesApp.dll  RazorPagesApp.runtimeconfig.json  appsettings.json  wwwroot
RazorPagesApp.deps.json  RazorPagesApp.pdb  appsettings.Development.json     web.config
```

Deploying the app to Docker Container

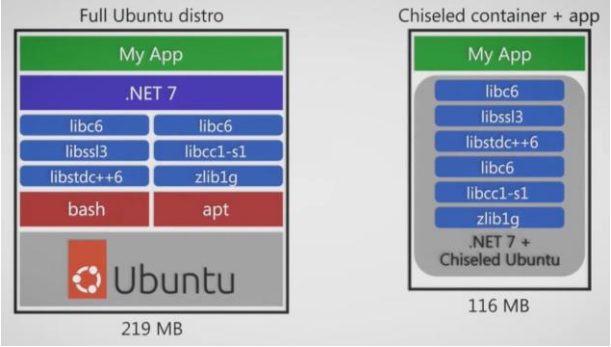
```
// Creating the Dockerfile
yeongseon@vm-ubuntu:~/RazorPagesApp$ cat Dockerfile
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build-env
WORKDIR /App

# Copy everything
COPY . ./
# Restore as distinct layers
RUN dotnet restore
# Build and publish a release
RUN dotnet publish -c Release -o out

# Build runtime image
FROM mcr.microsoft.com/dotnet/aspnet:6.0
WORKDIR /App
COPY --from=build-env /App/out .
ENTRYPOINT ["dotnet", "RazorPagesApp.dll"]
```

Deploying the app to Docker Container

```
// Creating Docker image
yeongseon@vm-ubuntu:~/RazorPagesApp$ sudo docker build -t razorpagesapp-image -f Dockerfile .
// Creating Docker container
yeongseon@vm-ubuntu:~/RazorPagesApp$ sudo docker create --name razorpagesapp razorpagesapp-image
062ee4dc8555baaae1959f4d7774fe56506b51bbddd9c7be2a187335dc89c51c

yeongseon@vm-ubuntu:~/RazorPagesApp$ sudo docker ps -a
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS    PORTS    NAMES
062ee4dc8555   razorpagesapp-image   "dotnet RazorPagesAp…"   28 seconds ago   Created          razorpagesapp
// Starting container
yeongseon@vm-ubuntu:~/RazorPagesApp$ sudo docker start razorpagesapp
// Stopping container
yeongseon@vm-ubuntu:~/RazorPagesApp$ sudo docker stop razorpagesapp
```

Deploying the app to Docker Container

- Minimal OCI images: .NET in Chiseled Ubuntu containers
  - Distroless images = Bare minimum
  - Quicker to pull/start
  - More secure – no shell, no package manager, non-root user



Deploy ASP.NET Core app to Docker Container

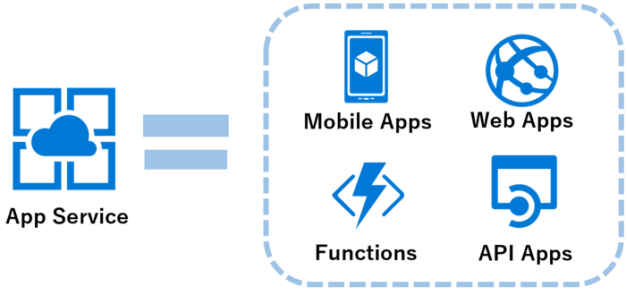
```
// Creating the Dockerfile.chiseled
yeongseon@vm-ubuntu:~/RazorPagesApp$ cat Dockerfile.chiseled
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build-env
WORKDIR /App

# Copy everything
COPY . ./
# Restore as distinct layers
RUN dotnet restore
# Build and publish a release
RUN dotnet publish -c Release -o out

# Build runtime image
FROM mcr.microsoft.com/dotnet/nightly/runtime:6.0-jammy-chiseledWORKDIR /App
COPY --from=build-env /App/out .0
ENTRYPOINT ["dotnet", "RazorPagesApp.dll"]
```

Deploy ASP.NET Core app to Docker Container

```
yeongseon@vm-ubuntu:~/RazorPagesApp$ sudo docker build -t razorpagesapp-image-chiseled -f
Dockerfile.chiseled .

yeongseon@vm-ubuntu:~/RazorPagesApp$ sudo docker images
REPOSITORY                          TAG             IMAGE ID      CREATED         SIZE
razorpagesapp-image-chiseled        latest          2fde7f446bb4  16 seconds ago  92MB
razorpagesapp-image                 latest          7b2b6daa77ef  5 days ago      216MB
mcr.microsoft.com/dotnet/sdk        6.0             0ff04a23cb9e  11 days ago     737MB
mcr.microsoft.com/dotnet/aspnet     6.0             33a9c85ed2f6  11 days ago     208MB
```

Deploy ASP.NET Core app to Azure App Service

- Azure App Service
  - A Fully managed platform as a service (PaaS)
- App Service on Linux



Deploy ASP.NET Core app to Azure App Service

```
// Sign into Azure account
az login

// Deploy the code
Az web up –sku F1 –name <app-name> --os-type <os>
```

Reference

https://learn.microsoft.com/en-us/dotnet/core/introduction

https://ubuntu.com/blog/install-dotnet-on-ubuntu

https://devblogs.microsoft.com/dotnet/dotnet-6-is-now-in-ubuntu-2204/

https://www.how2shout.com/linux/3-ways-to-install-net-6-dotnet-on-ubuntu-20-04-lts-focal-fossa/

ASP.NET Core fundamentals overview | Microsoft Learn

Thank you!