Pdf Vision .Net

(Multi-platform .Net library)

SautinSoft

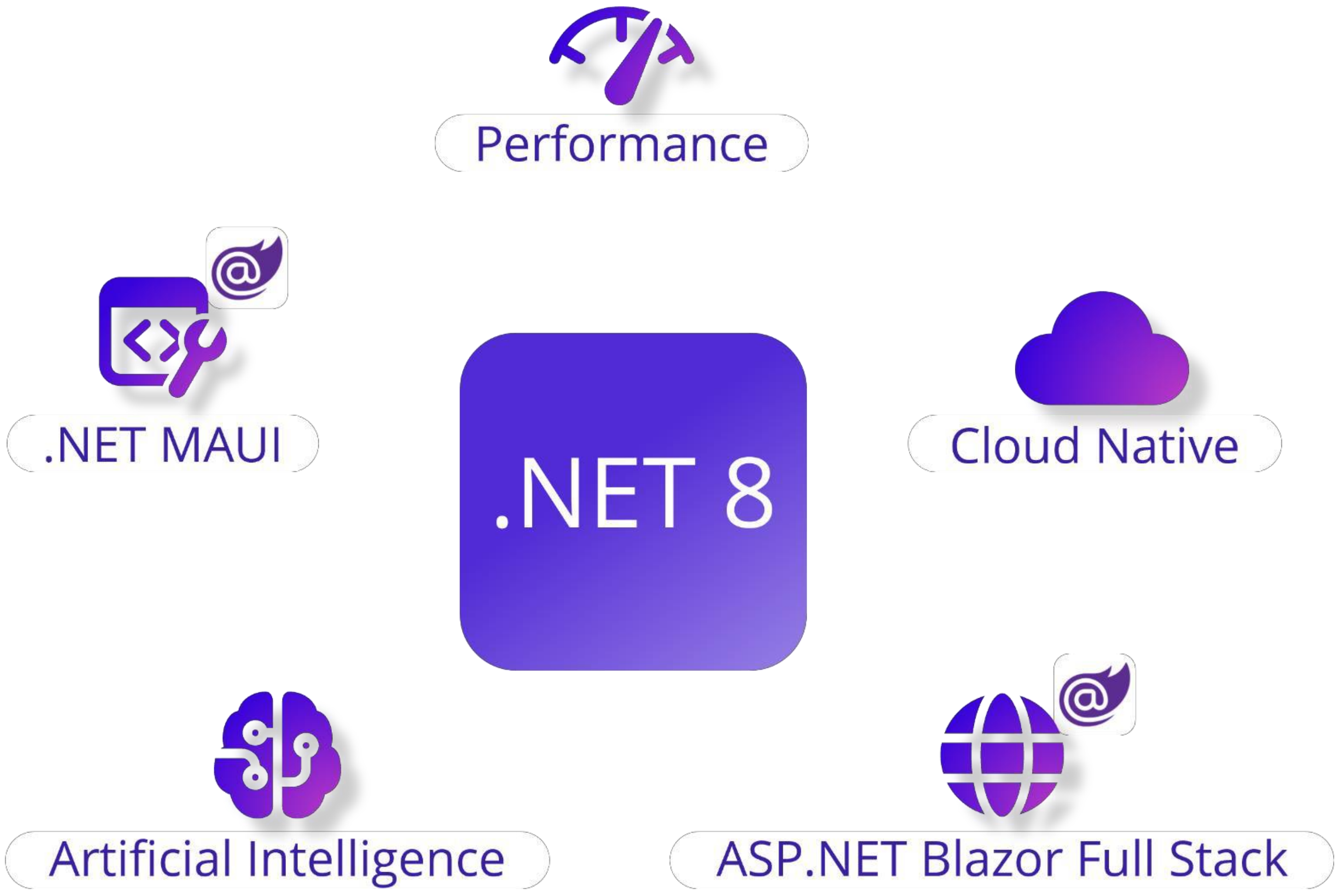
Linux development manual

Table of Contents

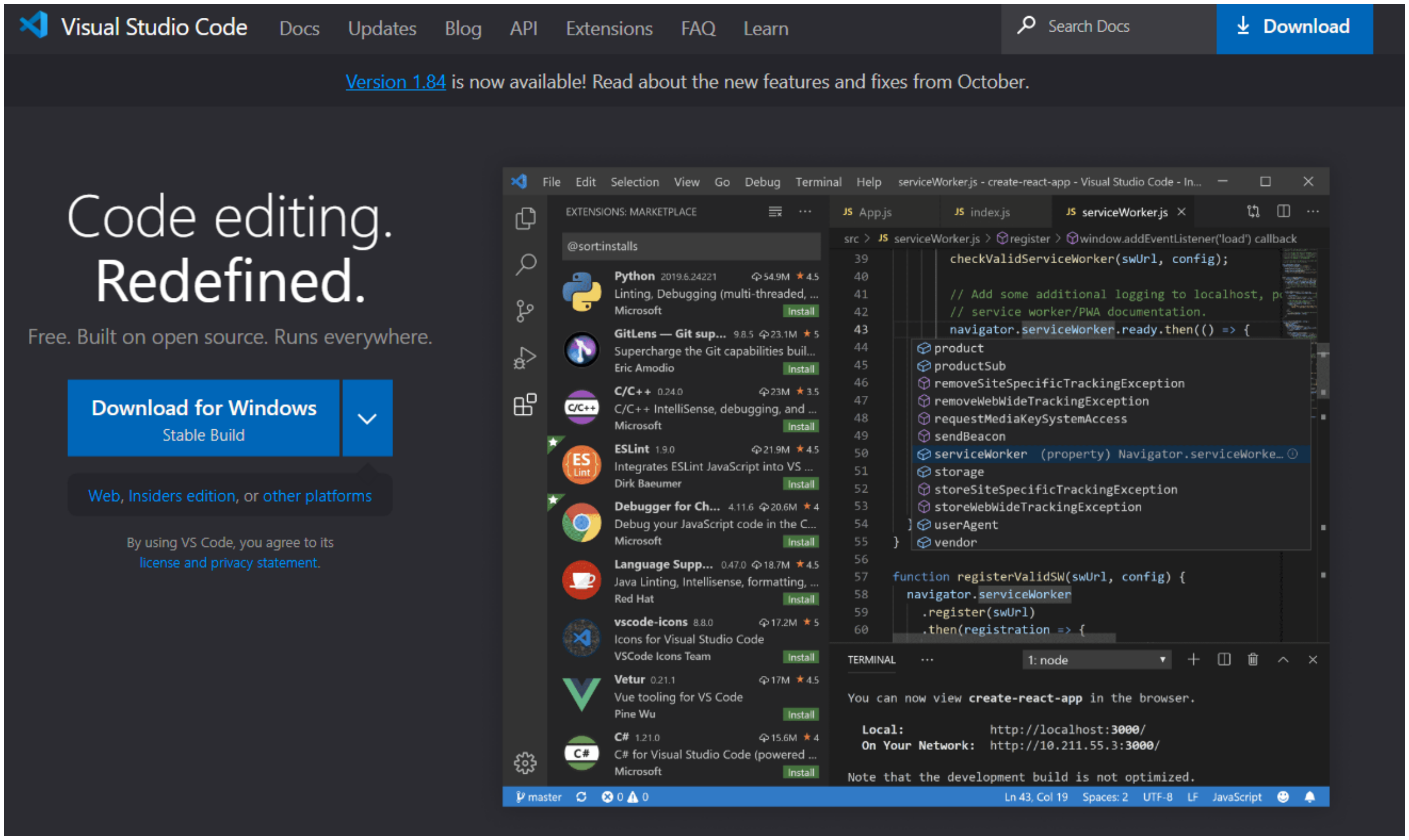
1. Preparing environment

In order to build multi-platform applications using .NET on Linux, the first steps are for installing in our Linux machine the required tools.

We need to install .NET SDK from Microsoft and to allow us to develop easier, we will install an advance editor with a lot of features, Visual Studio Code from Microsoft.
Both installations are very easy and the detailed description can be found by these two links:
Install .NET SDK for Linux.



Install VS Code for Linux.
Once installed VS Code, you need to install a C# extension to facilitate us to code and debugging:
Install C# extension.


Visual Studio Code
Docs
Updates
Blog
API
Extensions
FAQ
Learn
Search Docs
Download
File Edit Selection View Go Debug Terminal Help
serviceWorker.js - create-react-app - Visual Studio Code - In...
EXTENSIONS: MARKETPLACE
@sort:installs
App.js
index.js
serviceWorker.js
src > serviceWorker.js > register > window.addEventListener('load') callback
TERMINAL
1: node
master
Ln 43, Col 19 Spaces: 2 UTF-8 LF JavaScript

Version 1.84 is now available! Read about the new features and fixes from October.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows
Stable Build

Web, Insiders edition, or other platforms

By using VS Code, you agree to its
license and privacy statement.

Python 2019.6.24221 54.9M ★4.5
Linting, Debugging (multi-threaded, ...
Microsoft Install

GitLens — Git sup... 9.8.5 23.1M ★5
Supercharge the Git capabilities buil...
Eric Amodio Install

C/C++ 0.24.0 23M ★3.5
C/C++ IntelliSense, debugging, and ...
Microsoft Install

ESLint 1.9.0 21.9M ★4.5
Integrates ESLint JavaScript into VS ...
Dirk Baeumer Install

Debugger for Ch... 4.11.6 20.6M ★4
Debug your JavaScript code in the C...
Microsoft Install

Language Supp... 0.47.0 18.7M ★4.5
Java Linting, Intellisense, formatting, ...
Red Hat Install

vscode-icons 8.8.0 17.2M ★5
Icons for Visual Studio Code
VSCode Icons Team Install

Vetur 0.21.1 17M ★4.5
Vue tooling for VS Code
Pine Wu Install

C# 1.21.0 15.6M ★4
C# for Visual Studio Code (powered ...
Microsoft Install

```
        checkValidServiceWorker(swUrl, config);

        // Add some additional logging to localhost, p
        // service worker/PWA documentation.
        navigator.serviceWorker.ready.then(() => {
          product
          productSub
          removeSiteSpecificTrackingException
          removeWebWideTrackingException
          requestMediaKeySystemAccess
          sendBeacon
          serviceWorker  (property) Navigator.serviceWorke...
          storage
          storeSiteSpecificTrackingException
          storeWebWideTrackingException
      } userAgent
  }     vendor

function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
```

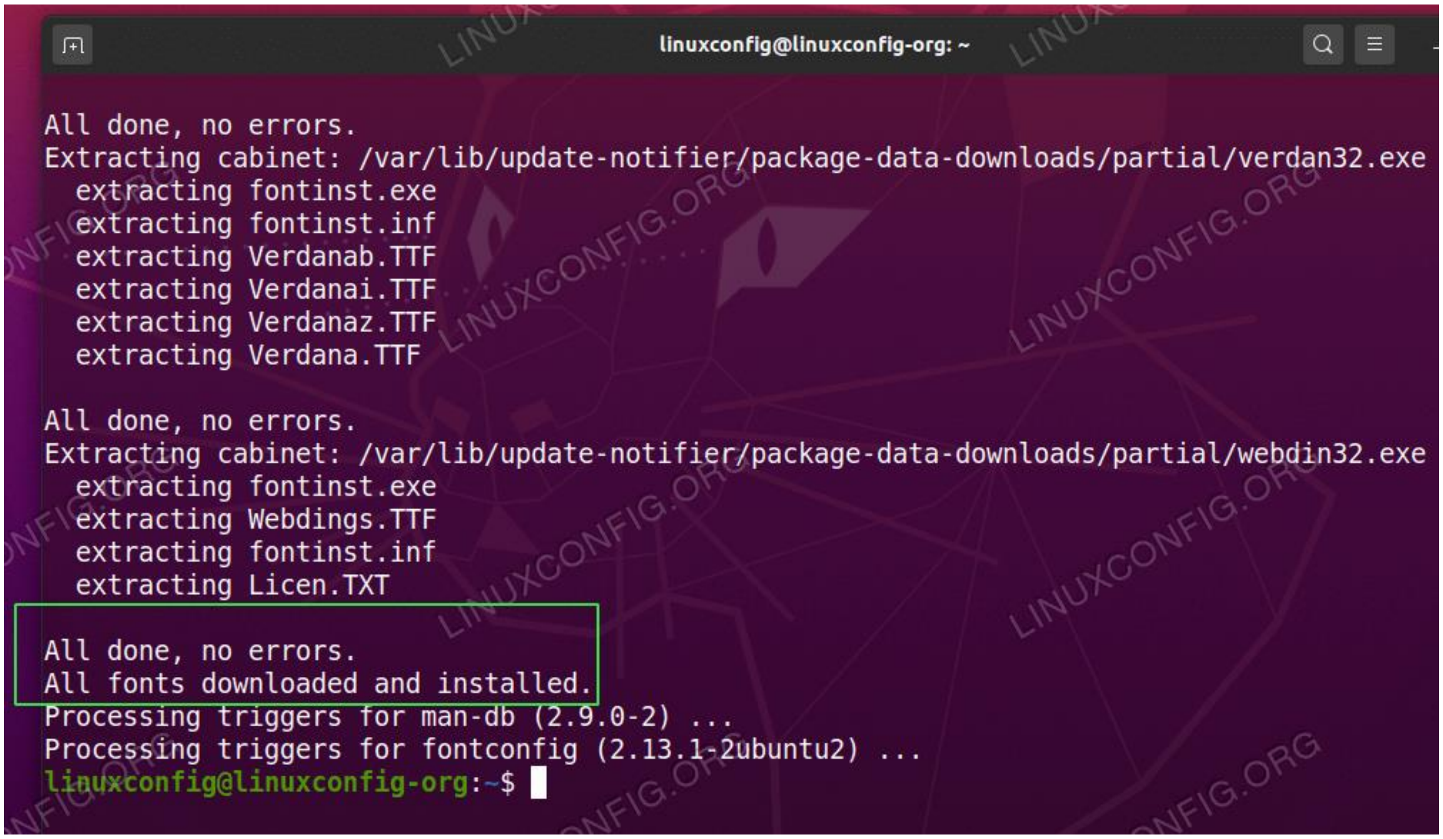
```
You can now view create-react-app in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://10.211.55.3:3000/

Note that the development build is not optimized.
```

1.1. Check the installed Fonts availability

Check that the directory with fonts "/usr/share/fonts/truetype" is exist. Also check that it contains *.ttf files.

If you don't see this folder, you may install “Microsoft TrueType core fonts” using terminal and command:

```
$ sudo apt install ttf-mscorefonts-installer
```



```
All done, no errors.
Extracting cabinet: /var/lib/update-notifier/package-data-downloads/partial/verdan32.exe
  extracting fontinst.exe
  extracting fontinst.inf
  extracting Verdanab.TTF
  extracting Verdanai.TTF
  extracting Verdanaz.TTF
  extracting Verdana.TTF

All done, no errors.
Extracting cabinet: /var/lib/update-notifier/package-data-downloads/partial/webdin32.exe
  extracting fontinst.exe
  extracting Webdings.TTF
  extracting fontinst.inf
  extracting Licen.TXT

All done, no errors.
All fonts downloaded and installed.
Processing triggers for man-db (2.9.0-2) ...
Processing triggers for fontconfig (2.13.1-2ubuntu2) ...
linuxconfig@linuxconfig-org:~$ 
```

Read more about TrueType Fonts and "How to install Microsoft fonts, How to update fonts cache files, How to confirm new fonts installation".

In next paragraphs we will explain in detail how to create simple console application. All of them are based on this VS Code guide:

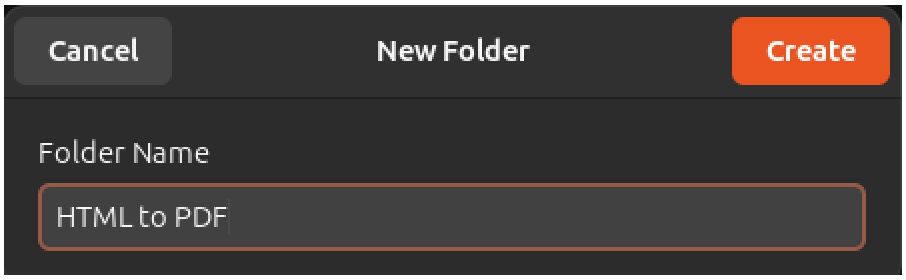
Get Started with C# and Visual Studio Code

Not only is possible to create .NET applications that will run on Linux using Linux as a developing platform. It is also possible to create it using a Windows machine and any modern Visual Studio version, as Microsoft Visual Studio Community 2022.

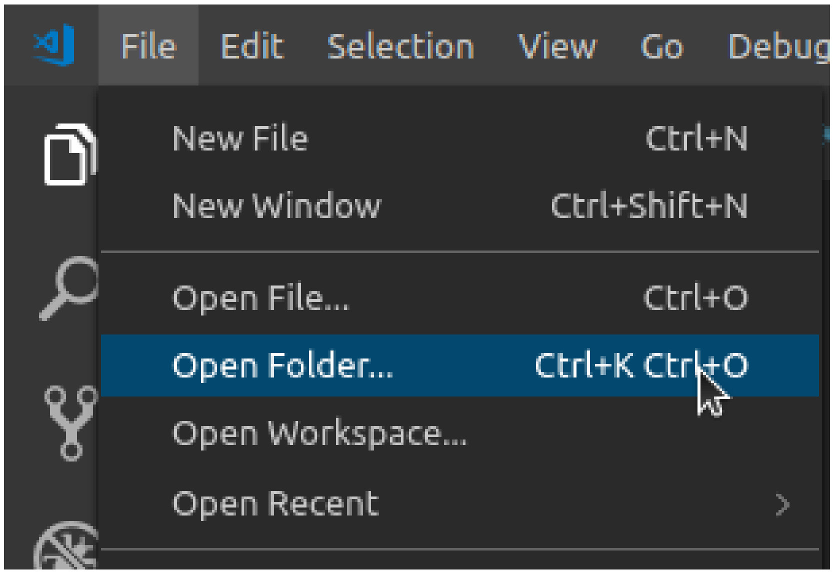
2. Creating "Convert HTML to PDF" application

Create a new folder in your Linux machine with the name ***HTML to PDF.***

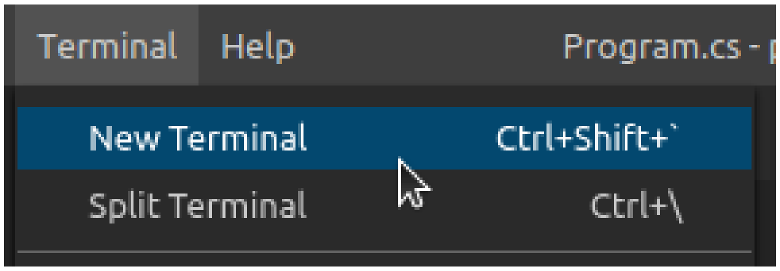
For example, let's create the folder "***HTML to PDF***" on Desktop (Right click-> New Folder):



Open VS Code and click in the menu ***File->Open Folder***. From the dialog, open the folder you've created previously:

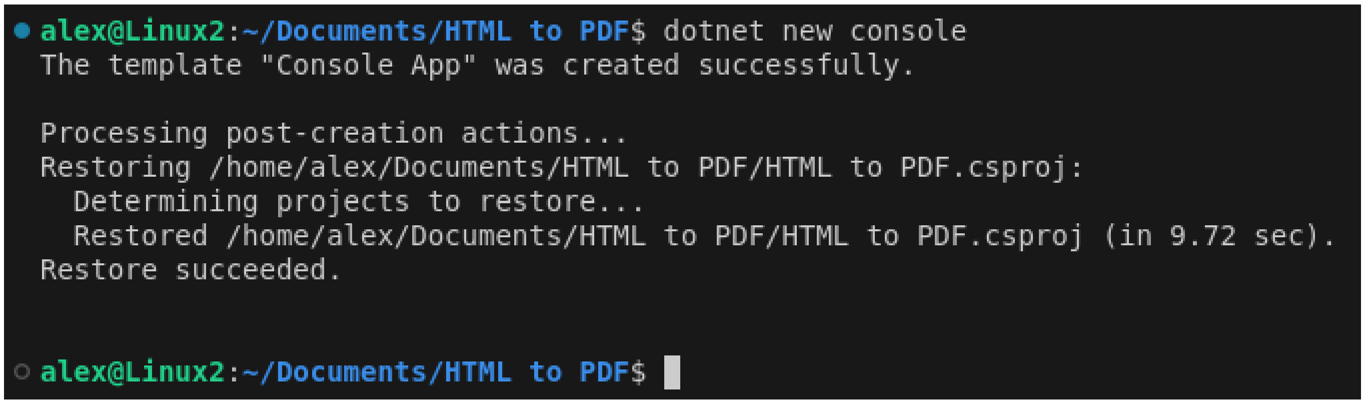


Now, open the integrated console – the Terminal: follow to the menu ***Terminal -> New Terminal*** (or press Ctrl+Shift+'):



Create a new console application, using ***dotnet*** command.

Type this command in the Terminal console: ***dotnet new console***



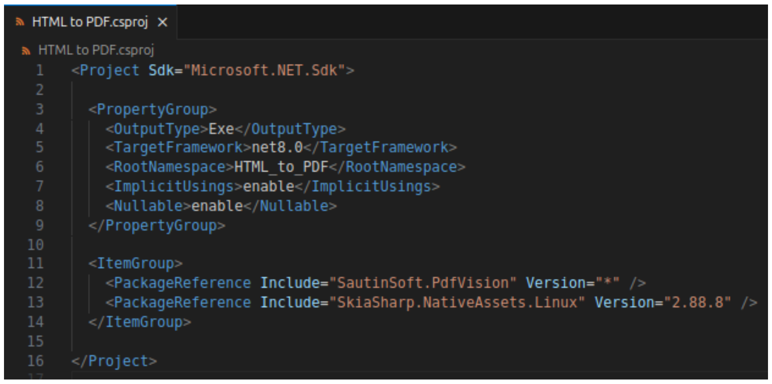
```
alex@Linux2:~/Documents/HTML to PDF$ dotnet new console
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring /home/alex/Documents/HTML to PDF/HTML to PDF.csproj:
  Determining projects to restore...
  Restored /home/alex/Documents/HTML to PDF/HTML to PDF.csproj (in 9.72 sec).
Restore succeeded.


alex@Linux2:~/Documents/HTML to PDF$ 
```

Now we are going to modify this simple application into an application that will convert a HTML file to a PDF file.

First of all, we need to add the package reference to the ***sautinsoft.pdfvision*** assembly using Nuget or the library SautinSoft.PdfVision.dll with additional references.

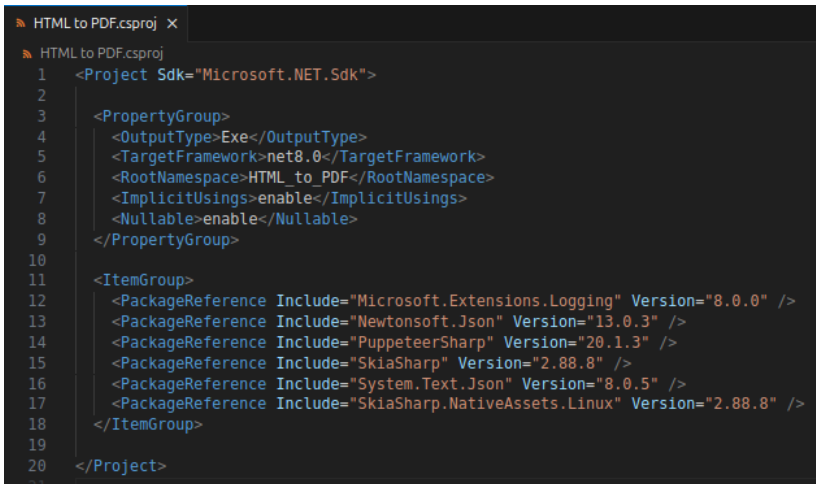
In order to do it, follow to the ***Explorer*** and open project file "***HTML to PDF.csproj***" :

In the first case (NuGet):



```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <RootNamespace>HTML_to_PDF</RootNamespace>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="SautinSoft.PdfVision" Version="*" />
    <PackageReference Include="SkiaSharp.NativeAssets.Linux" Version="2.88.8" />
  </ItemGroup>

</Project>
```

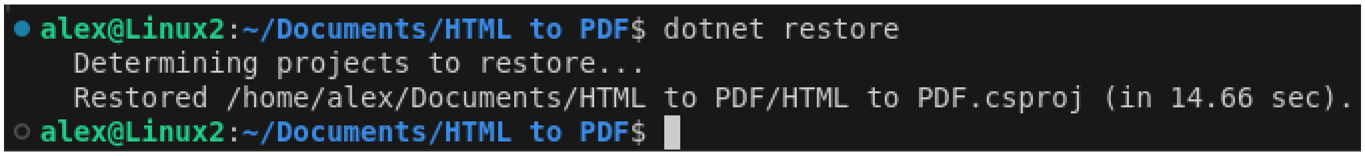
In the second case (SautinSoft.PdfVision.dll):



```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <RootNamespace>HTML_to_PDF</RootNamespace>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.Extensions.Logging" Version="8.0.0" />
    <PackageReference Include="Newtonsoft.Json" Version="13.0.3" />
    <PackageReference Include="PuppeteerSharp" Version="20.1.3" />
    <PackageReference Include="SkiaSharp" Version="2.88.8" />
    <PackageReference Include="System.Text.Json" Version="8.0.5" />
    <PackageReference Include="SkiaSharp.NativeAssets.Linux" Version="2.88.8" />
  </ItemGroup>

</Project>
```

At once as we've added the package references, we have to save the "***HTML to PDF.csproj***" and restore the added packages.

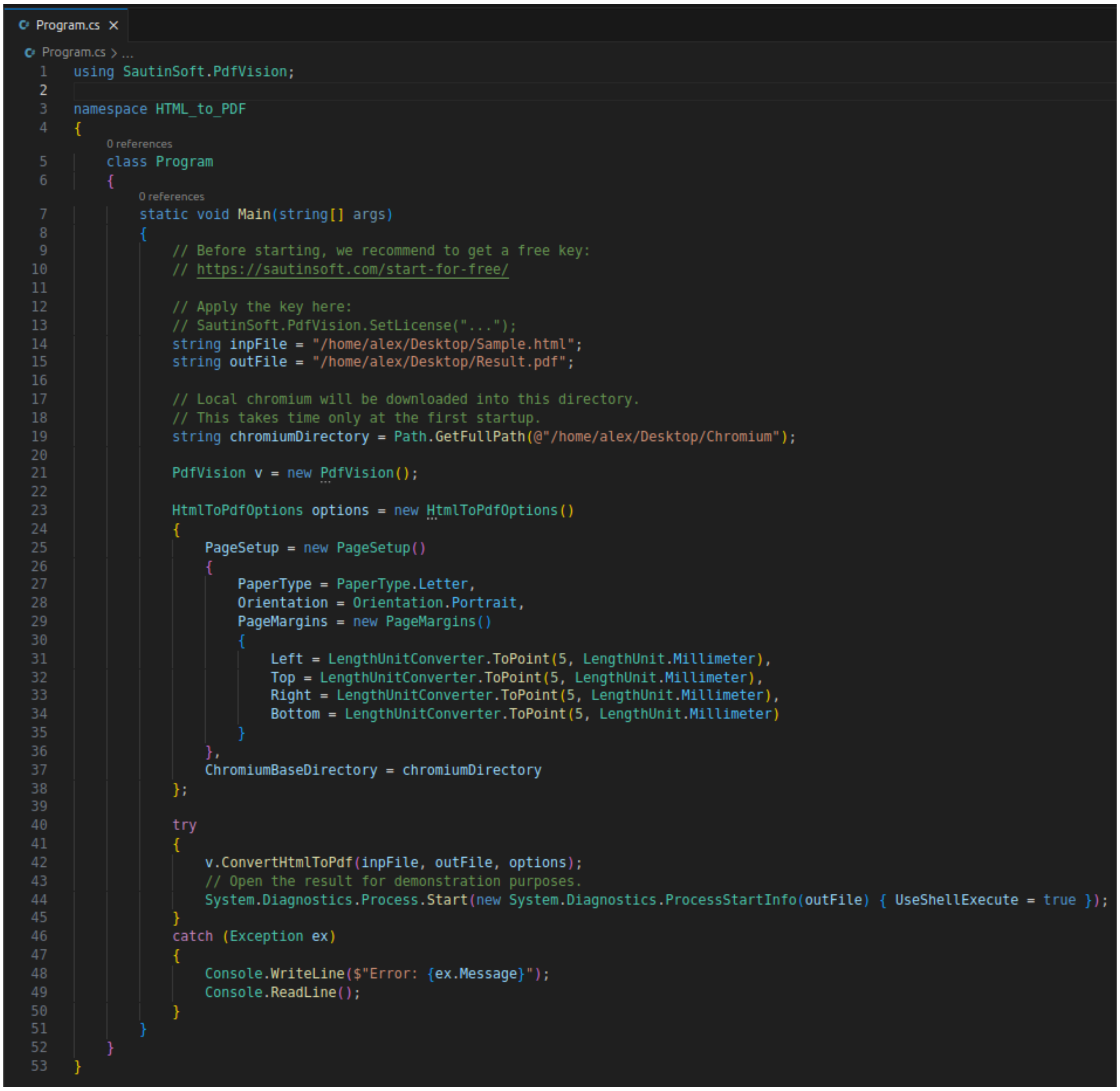
Follow to the ***Terminal*** and type the command: ***dotnet restore***



```
● alex@Linux2:~/Documents/HTML to PDF$ dotnet restore
   Determining projects to restore...
   Restored /home/alex/Documents/HTML to PDF/HTML to PDF.csproj (in 14.66 sec).
○ alex@Linux2:~/Documents/HTML to PDF$ 
```

Good, now our application has all the references and we can write the code to convert HTML to PDF.

Follow to the ***Explorer***, open the ***Program.cs***, remove all the code and type the new:



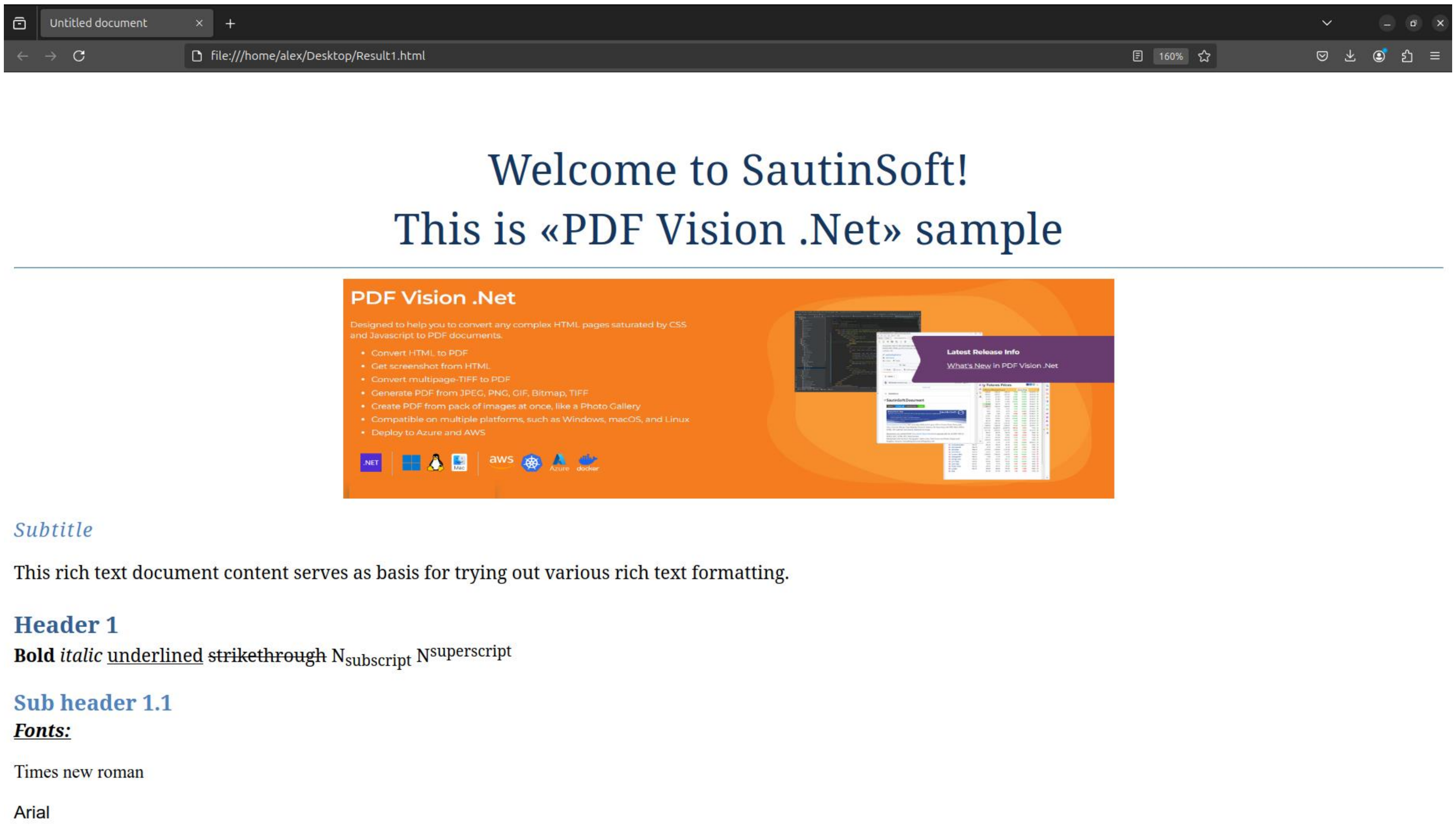
```csharp
using SautinSoft.PdfVision;

namespace HTML_to_PDF
{
    class Program
    {
        static void Main(string[] args)
        {
            // Before starting, we recommend to get a free key:
            // https://sautinsoft.com/start-for-free/

            // Apply the key here:
            // SautinSoft.PdfVision.SetLicense("...");
            string inpFile = "/home/alex/Desktop/Sample.html";
            string outFile = "/home/alex/Desktop/Result.pdf";

            // Local chromium will be downloaded into this directory.
            // This takes time only at the first startup.
            string chromiumDirectory = Path.GetFullPath(@"/home/alex/Desktop/Chromium");

            PdfVision v = new PdfVision();

            HtmlToPdfOptions options = new HtmlToPdfOptions()
            {
                PageSetup = new PageSetup()
                {
                    PaperType = PaperType.Letter,
                    Orientation = Orientation.Portrait,
                    PageMargins = new PageMargins()
                    {
                        Left = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
                        Top = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
                        Right = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
                        Bottom = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter)
                    }
                },
                ChromiumBaseDirectory = chromiumDirectory
            };

            try
            {
                v.ConvertHtmlToPdf(inpFile, outFile, options);
                // Open the result for demonstration purposes.
                System.Diagnostics.Process.Start(new System.Diagnostics.ProcessStartInfo(outFile) { UseShellExecute = true });
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
                Console.ReadLine();
            }
        }
    }
}
```

The code:

```
using SautinSoft.PdfVision;

namespace HTML_to_PDF
{
    class Program
    {
        static void Main(string[] args)
        {
            // Before starting, we recommend to get a free key:
            // https://sautinsoft.com/start-for-free/

            // Apply the key here:
            // SautinSoft.PdfVision.SetLicense("...");
            string inpFile = "/home/alex/Desktop/Sample.html";
            string outFile = "/home/alex/Desktop/Result.pdf";

            // Local chromium will be downloaded into this directory.
            // This takes time only at the first startup.
            string chromiumDirectory = Path.GetFullPath(@"/home/alex/Desktop/Chromium/");

            PdfVision v = new PdfVision();

            HtmlToPdfOptions options = new HtmlToPdfOptions()
            {
                PageSetup = new PageSetup()
                {
                    PaperType = PaperType.Letter,
                    Orientation = Orientation.Portrait,
                    PageMargins = new PageMargins()
                    {
                        Left = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
                        Top = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
                        Right = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter),
                        Bottom = LengthUnitConverter.ToPoint(5, LengthUnit.Millimeter)
                    }
                },
                ChromiumBaseDirectory = chromiumDirectory
            };

            try
            {
                v.ConvertHtmlToPdf(inpFile, outFile, options);
                // Open the result for demonstration purposes.
                System.Diagnostics.Process.Start(new
                System.Diagnostics.ProcessStartInfo(outFile) { UseShellExecute = true });
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
                Console.ReadLine();
            }
        }
    }
}
```

To make tests, we need an input HTML document. For our tests, let’s place a HTML file with the name “Sample.html” at the Desktop.

If we open this file in the default HTML browser, we’ll see its contents:



Welcome to SautinSoft!
This is «PDF Vision .Net» sample

PDF Vision .Net

Designed to help you to convert any complex HTML pages saturated by CSS and Javascript to PDF documents.

- Convert HTML to PDF
- Get screenshot from HTML
- Convert multipage-TIFF to PDF
- Generate PDF from JPEG, PNG, GIF, Bitmap, TIFF
- Create PDF from pack of images at once, like a Photo Gallery
- Compatible on multiple platforms, such as Windows, macOS, and Linux
- Deploy to Azure and AWS

Subtitle

This rich text document content serves as basis for trying out various rich text formatting.

Header 1

Bold *italic* underlined ~~strikethrough~~ $\text{N}_{\text{subscript}}$ $\text{N}^{\text{superscript}}$

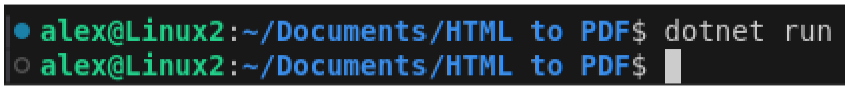
Sub header 1.1

Fonts:

Times new roman

Arial

Launch our application and convert the “Sample.html” into “Result.pdf”, type the command: ***dotnet run***

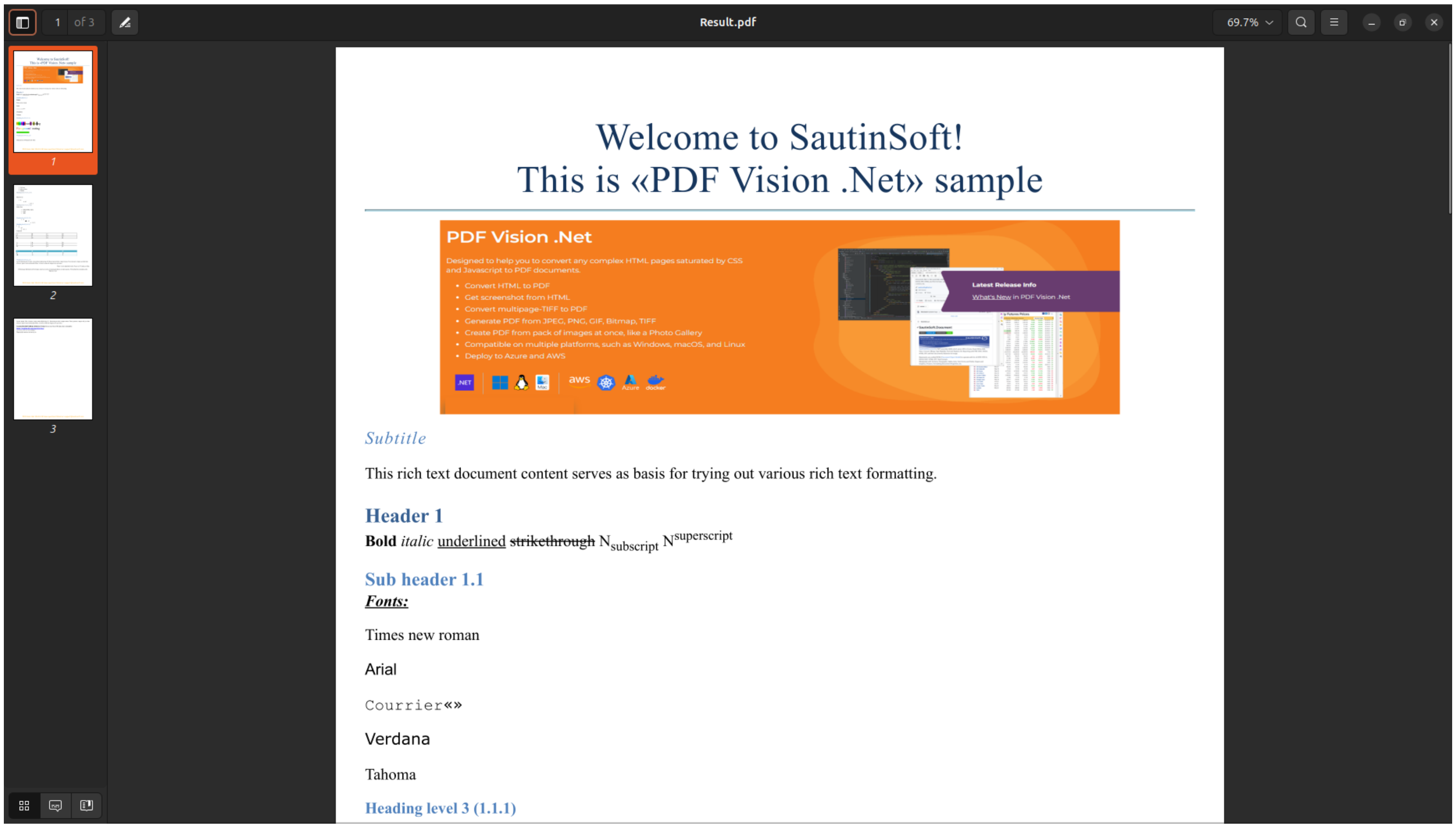


```
● alex@Linux2:~/Documents/HTML to PDF$ dotnet run
○ alex@Linux2:~/Documents/HTML to PDF$ 
```

If you don't see any exceptions, everything is fine and we can check the result produced by ehe PDF Vision .Net library.

The new file “Result.pdf” has to appear on the Desktop:

If we open this file in the default PDF Viewer, we'll see its contents:



Welcome to SautinSoft!
This is «PDF Vision .Net» sample

PDF Vision .Net

Designed to help you to convert any complex HTML pages saturated by CSS and Javascript to PDF documents.

- Convert HTML to PDF
- Get screenshot from HTML
- Convert multipage-TIFF to PDF
- Generate PDF from JPEG, PNG, GIF, Bitmap, TIFF
- Create PDF from pack of images at once, like a Photo Gallery
- Compatible on multiple platforms, such as Windows, macOS, and Linux
- Deploy to Azure and AWS

Latest Release Info
What's New in PDF Vision .Net

Subtitle

This rich text document content serves as basis for trying out various rich text formatting.

Header 1

Bold *italic* underlined ~~strikethrough~~ $\text{N}_{\text{subscript}}$ $\text{N}^{\text{superscript}}$

Sub header 1.1

Fonts:

Times new roman

Arial

Courrier«»

Verdana

Tahoma

Heading level 3 (1.1.1)

Well done! You have created the "HTML to PDF" application under Linux!

If you have any troubles or need extra code, or help, don't hesitate to ask our SautinSoft Team at support@sautinsoft.com!