

PDF .Net

(Multi-platform .Net library)

[SautinSoft](http://www.sautinsoft.com)

Linux development manual

Table of Contents

1. Preparing environment	2
1.1. Check the installed Fonts availability	3
2. Creating "Merge PDF files" application.....	6

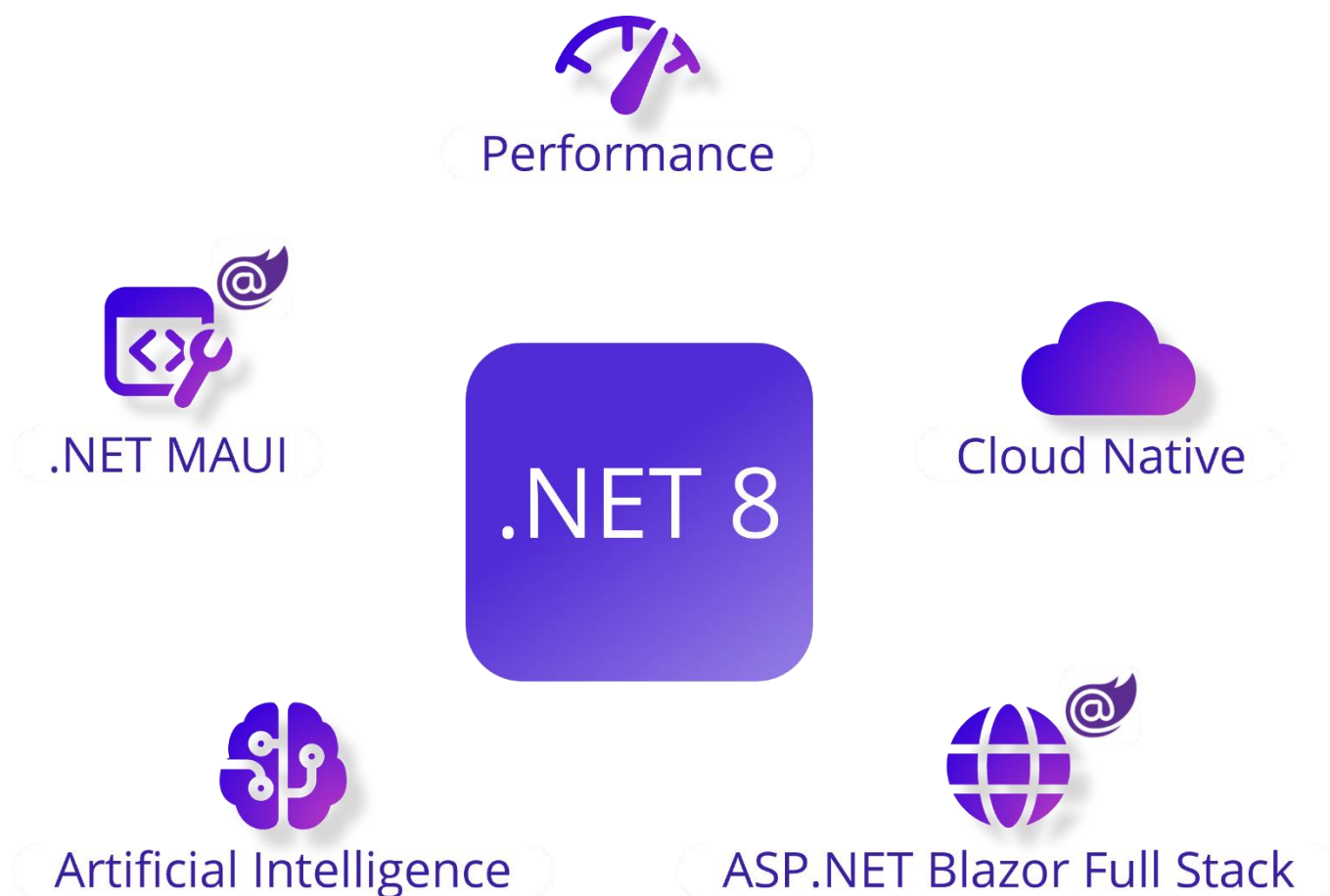
1. Preparing environment

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

We need to install .NET Core 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:

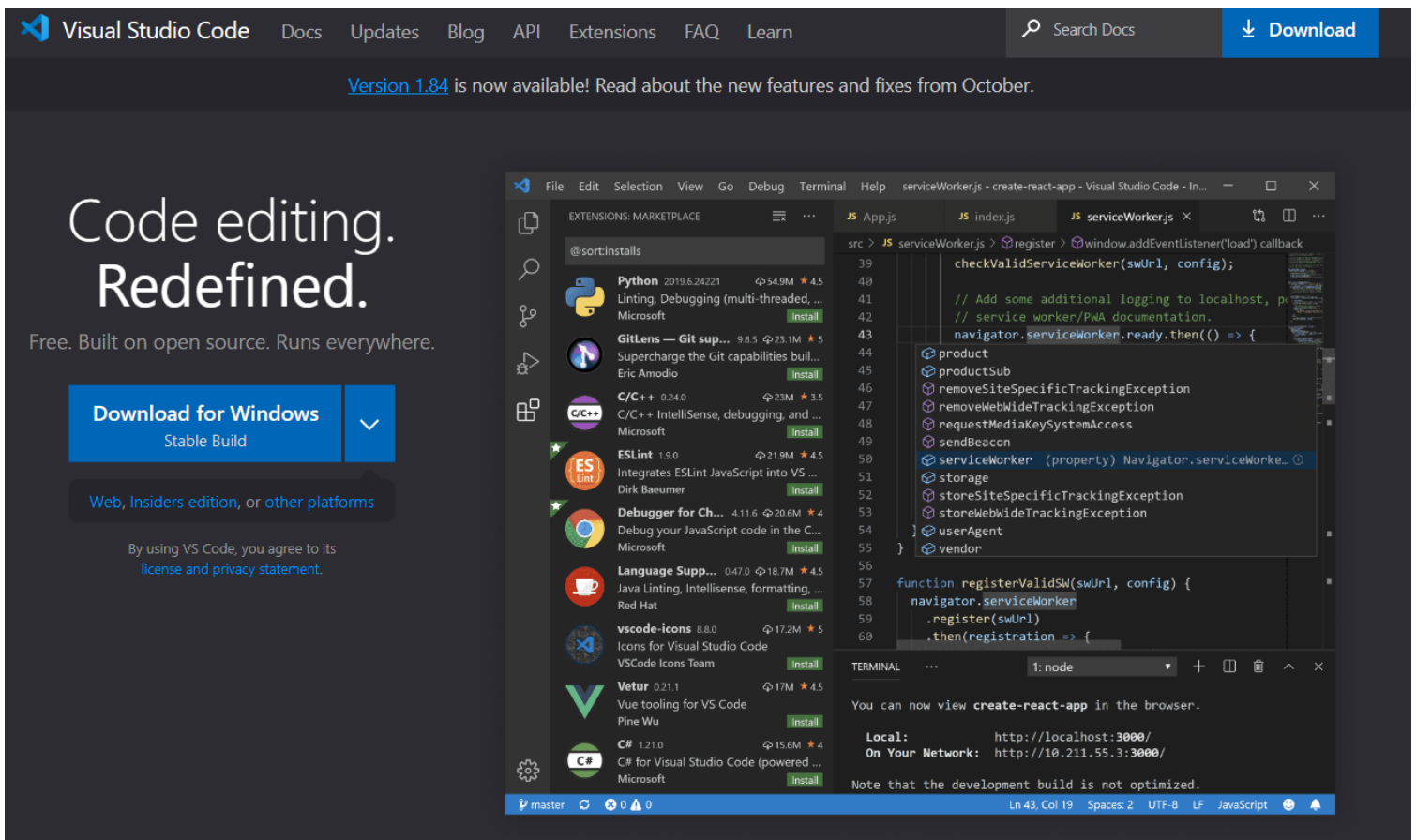
[Install .NET Core SDK for Linux.](#)



[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](#).



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, make these steps:

1. Download the archive with `*.ttf` fonts: <https://sautinsoft.com/components/fonts.tar>
2. Uncompress the downloaded font's archive to a directory and add it to the font path, a list of directories containing fonts:

```
# tar xvzf
```

3. Create a directory for new fonts

```
# mkdir /usr/share/fonts/truetype
```

4. Move the uncompressed font files to the new font directory

```
# mv *.ttf /usr/share/fonts/truetype
```

5. Navigate to the font directory

```
# cd /usr/share/fonts/truetype
```

6. Create fonts.scale and fonts.dir

```
# mkfontscale && mkfontdir  
# fc-cache
```

7. Add the new font directory to the X11 font path

```
# chkfontpath --add /usr/share/fonts/truetype
```

8. Restart X font server

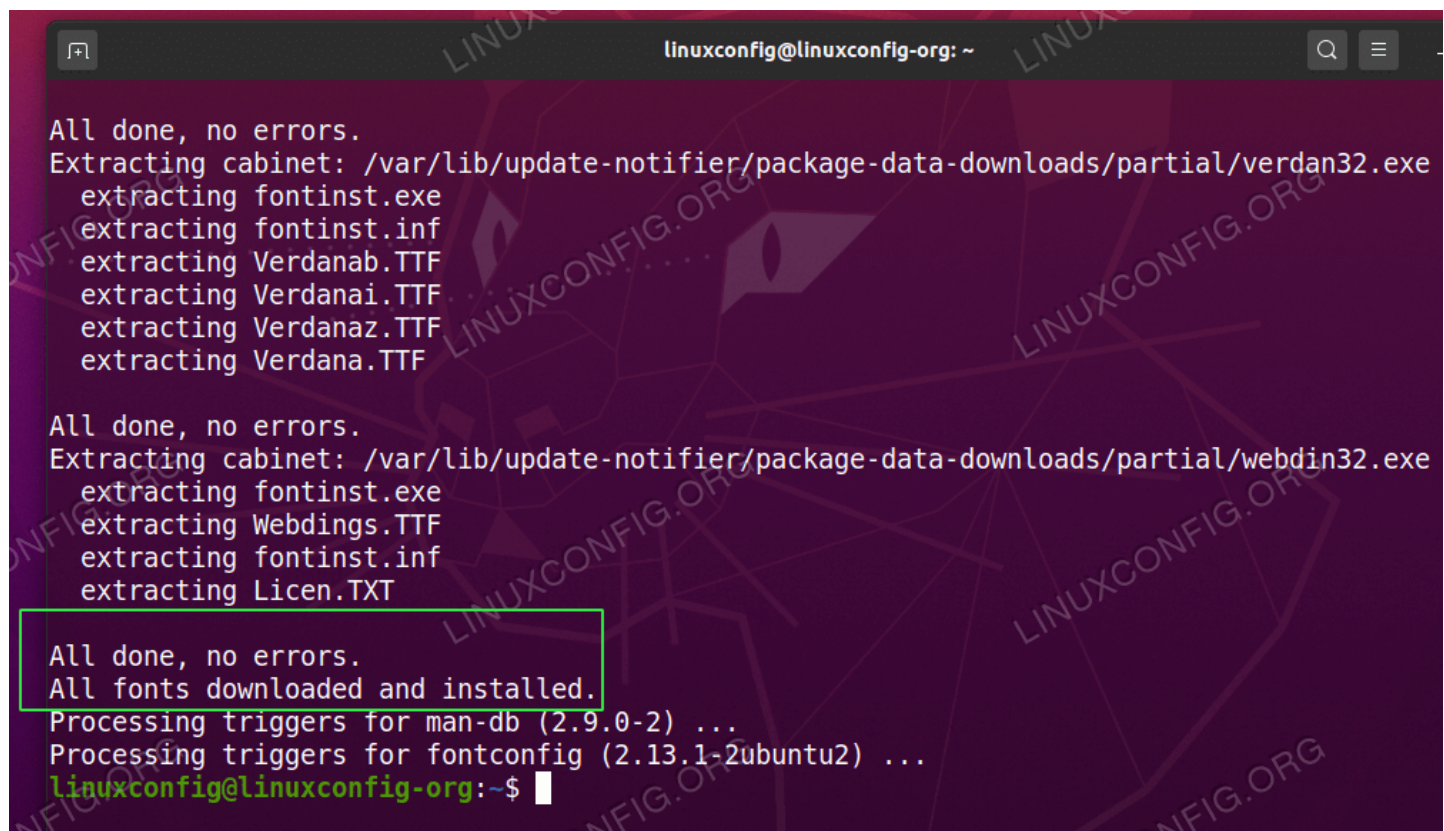
```
# /etc/rc.d/init.d/xfs restart
```

You can verify the successful addition of the new path by running `chkfontpath` command or by listing X font server's `/etc/X11/XF86Config` file.

If you do not have root access, copy the *.ttf to `~/.fonts` directory instead.

Or you may install "Microsoft TrueType core fonts" using terminal and command:

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



```
linuxconfig@linuxconfig-org: ~  
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 Verdanz.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"](#) .

With these steps, we will ready to start developing.

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

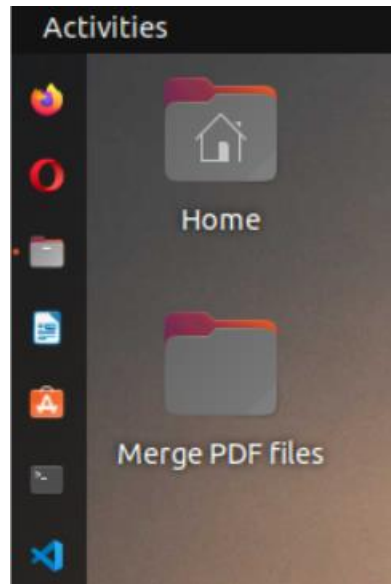
[Get Started with C# and Visual Studio Code](#)

Not only is possible to create .NET Core 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.

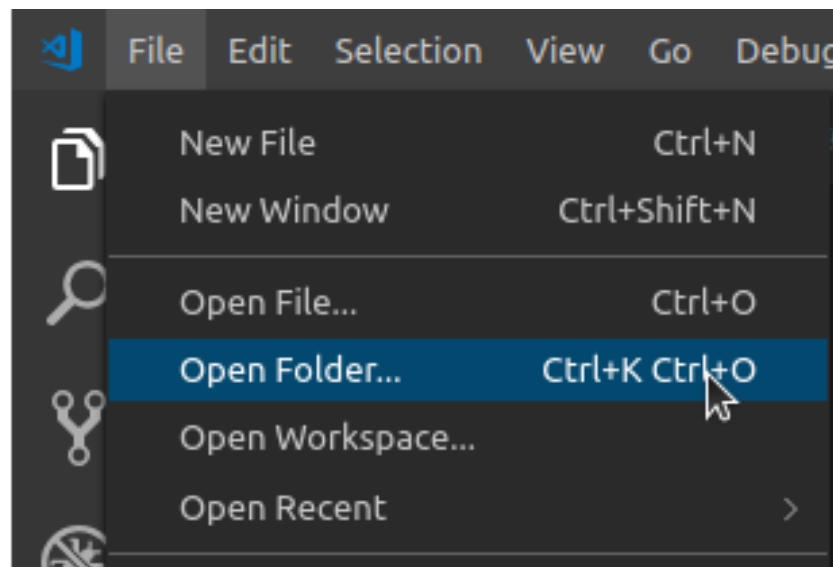
2. Creating “Merge PDF files” application

Create a new folder in your Linux machine with the name **Merge PDF Files**.

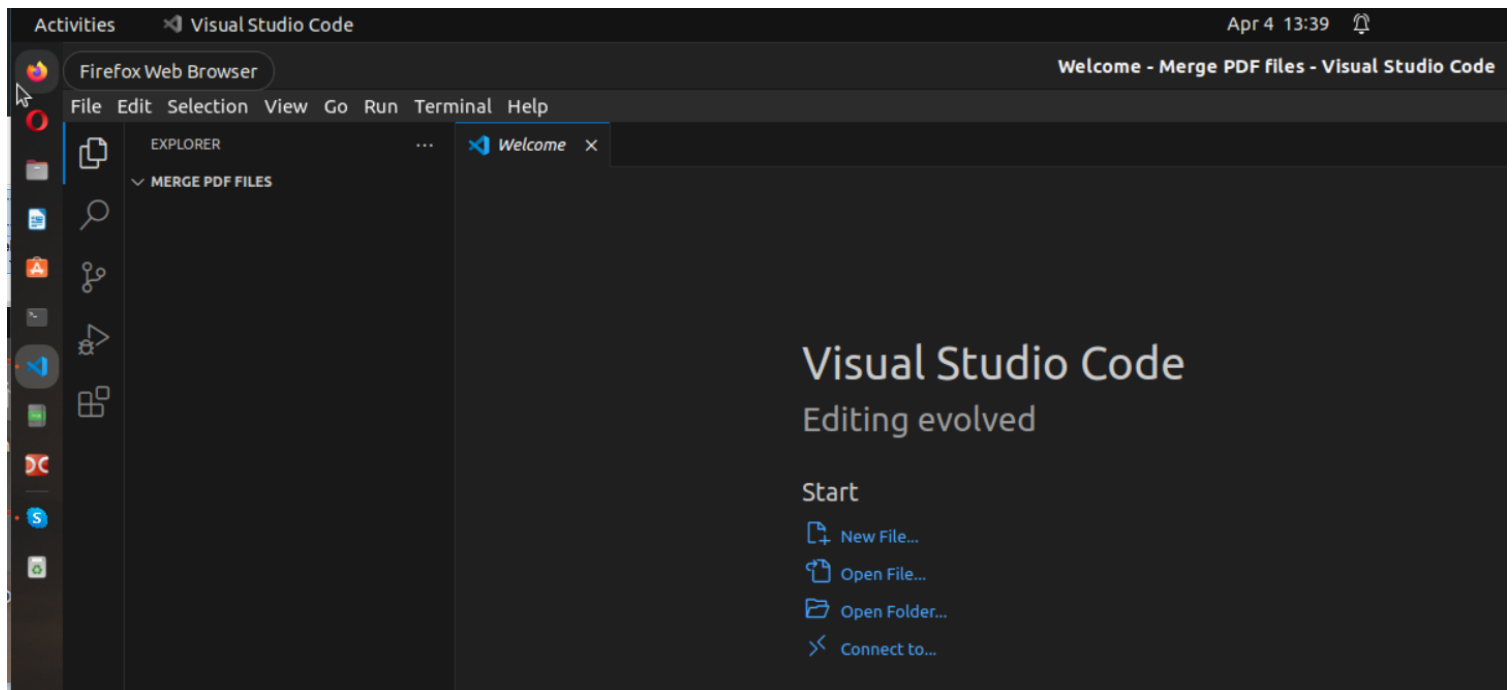
For example, let's create the folder “**Merge PDF Files**” on Desktop (Right click-> New Folder):



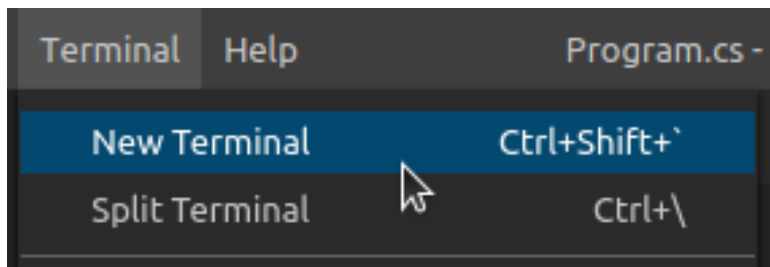
Open VS Code and click in the menu **File->Open Folder**. From the dialog, open the folder you've created previously:



Next you will see the similar screen:

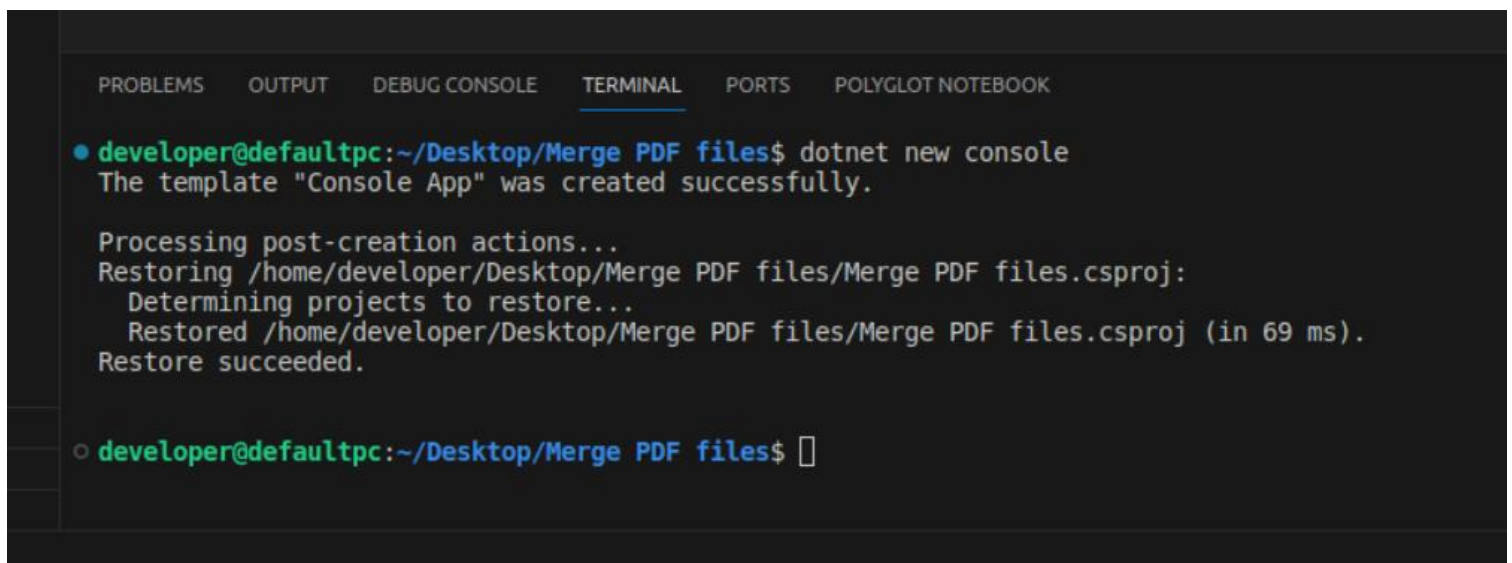


Now, open the integrated console – the Terminal: follow to the menu **Terminal** -> **New Terminal** (or press Ctrl+Shift+`):

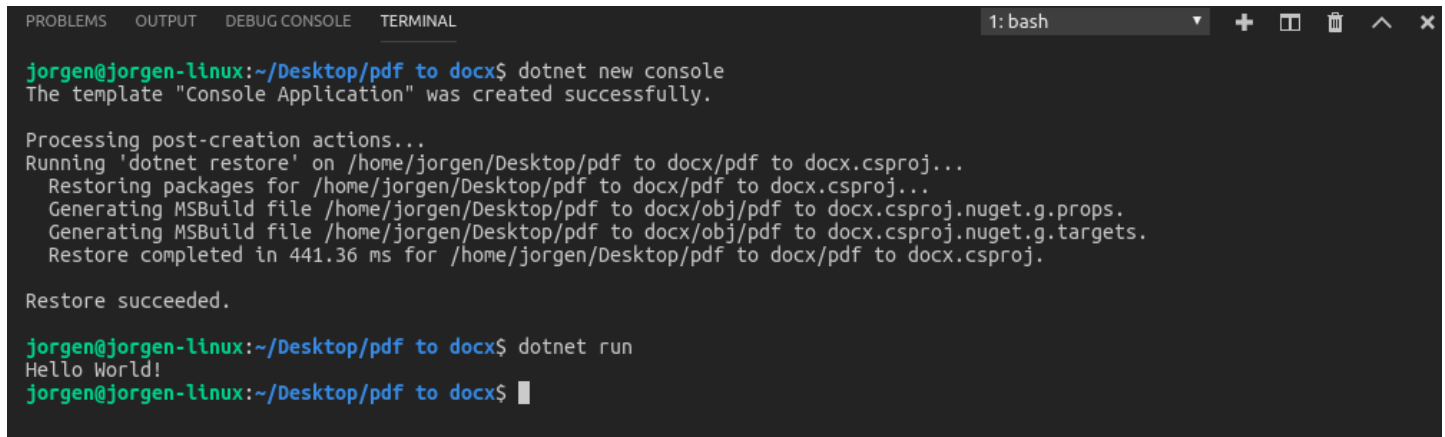


Create a new console application, using **dotnet** command.

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



A new simple ***Hello world!*** console application has been created. To execute it, type this command: ***dotnet run***



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
jorgen@jorgen-linux:~/Desktop/pdf to docx$ dotnet new console
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on /home/jorgen/Desktop/pdf to docx/pdf to docx.csproj...
  Restoring packages for /home/jorgen/Desktop/pdf to docx/pdf to docx.csproj...
  Generating MSBuild file /home/jorgen/Desktop/pdf to docx/obj/pdf to docx.csproj.nuget.g.props.
  Generating MSBuild file /home/jorgen/Desktop/pdf to docx/obj/pdf to docx.csproj.nuget.g.targets.
  Restore completed in 441.36 ms for /home/jorgen/Desktop/pdf to docx/pdf to docx.csproj.

Restore succeeded.

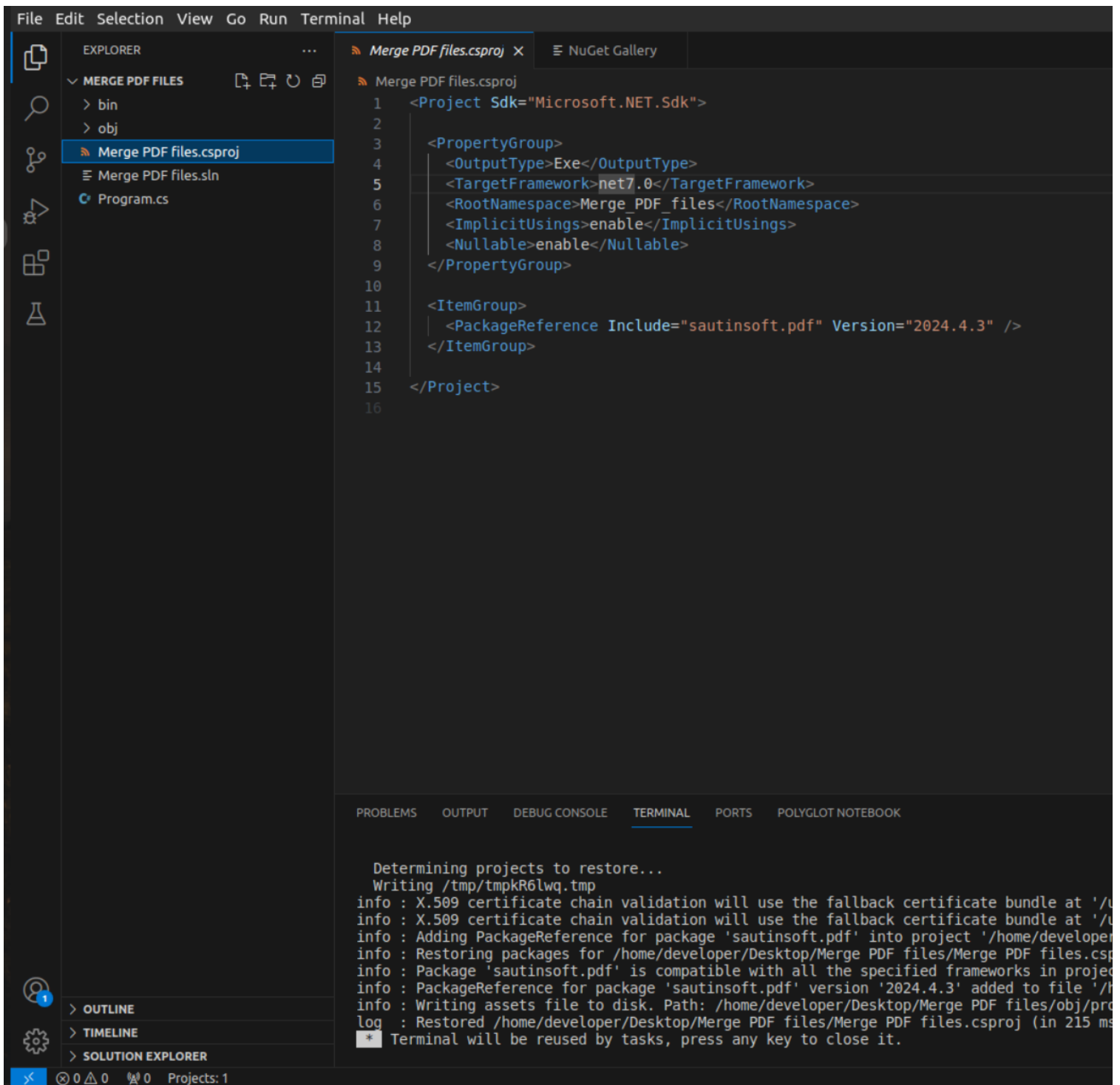
jorgen@jorgen-linux:~/Desktop/pdf to docx$ dotnet run
Hello World!
jorgen@jorgen-linux:~/Desktop/pdf to docx$
```

You can see the typical “Hello world!” message.

Now we are going to convert this simple application into something more interesting. We’ll transform it into an application that will merge a few PDF files in a big one PDF document.

First of all, we need to add the package reference to the ***sautinsoft.pdf*** assembly using Nuget.

In order to do it, follow to the ***Explorer*** and open project file “***Merge PDF Files.csproj***” within VS Code to edit it:



Add these lines into the file "**Merge PDF Files.csproj**":

```
<ItemGroup>
```

```
<PackageReference Include="Pkcs11Interop" Version="5.1.2" />
<PackageReference Include="Portable.BouncyCastle" Version="1.9.0" />
<PackageReference Include="SkiaSharp" Version="2.88.7" />
<PackageReference Include="SkiaSharp.HarfBuzz" Version="2.88.7" />
<PackageReference Include="Svg.Skia" Version="1.0.0.18" />
<PackageReference Include="System.IO.Packaging" Version="4.4.0" />
<PackageReference Include="System.Text.Encoding.CodePages" Version="4.5.0" />
<PackageReference Include="System.Xml.XPath.XmlDocument" Version="4.3.0" />
```

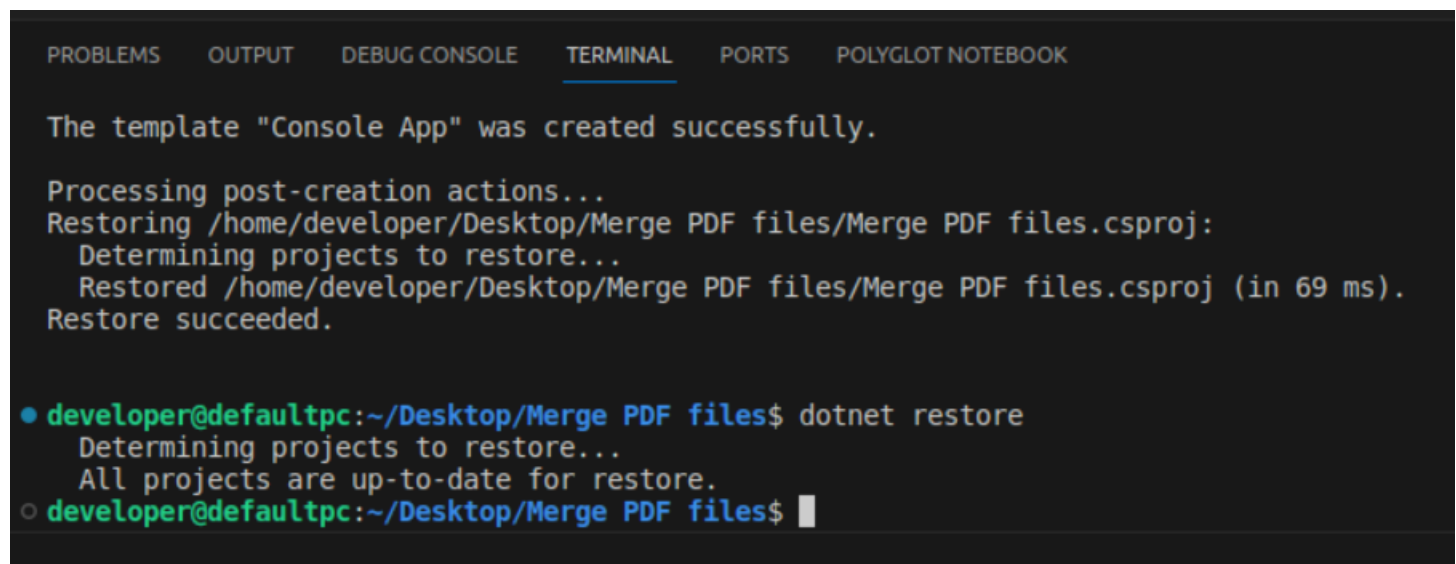
It's the reference to **sautinsoft.pdf** package from Nuget.

At the moment of writing this manual, the latest version of **sautinsoft.pdf** was 2024.X. But you may specify the latest version, to know what is the latest, follow:

<https://www.nuget.org/packages/sautinsoft.pdf/>

At once as we've added the package reference, we have to save the "**Merge PDF Files.csproj**" and restore the added package.

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



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POLYGLOT NOTEBOOK

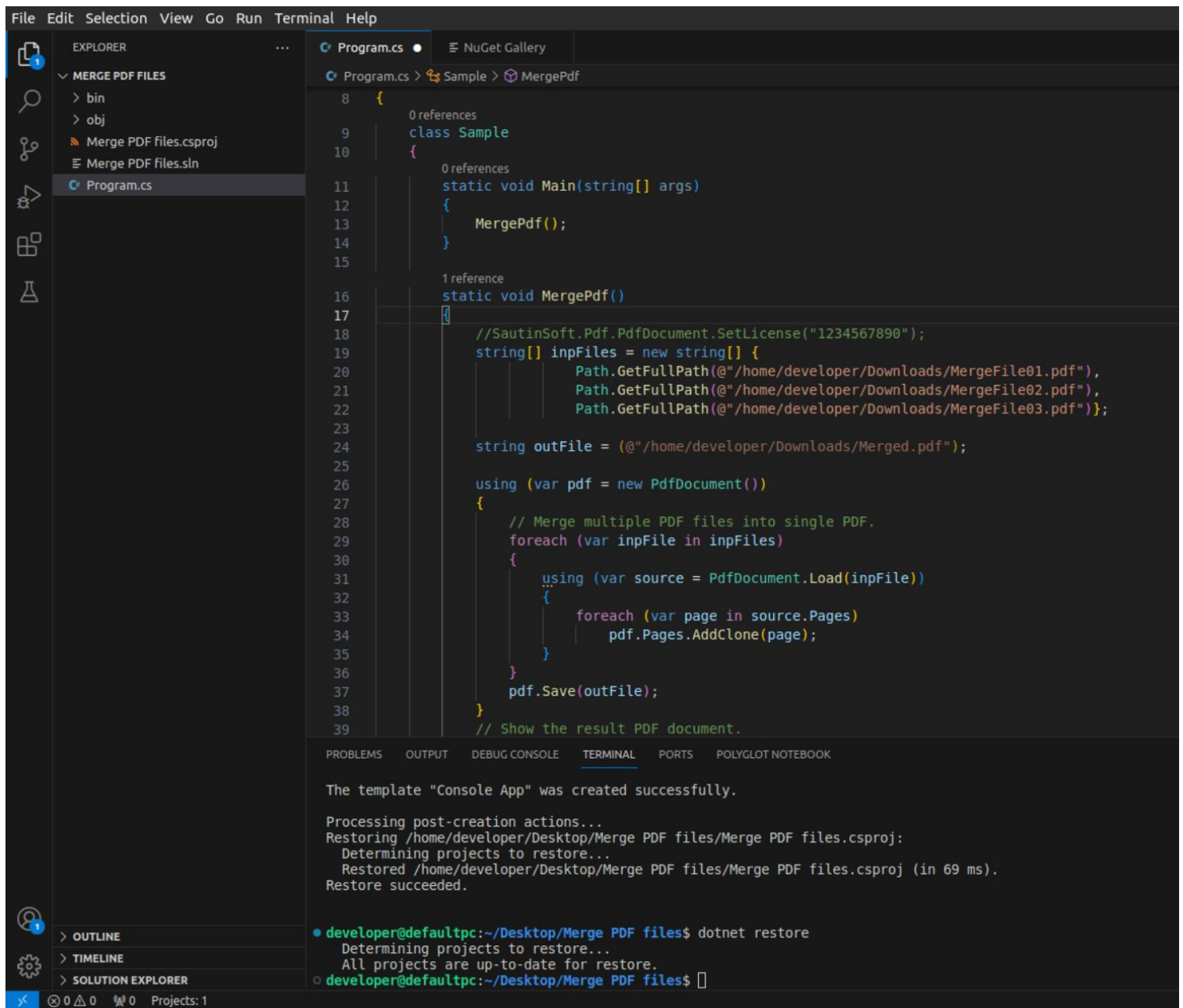
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring /home/developer/Desktop/Merge PDF files/Merge PDF files.csproj:
  Determining projects to restore...
  Restored /home/developer/Desktop/Merge PDF files/Merge PDF files.csproj (in 69 ms).
Restore succeeded.

● developer@defaultpc:~/Desktop/Merge PDF files$ dotnet restore
  Determining projects to restore...
  All projects are up-to-date for restore.
○ developer@defaultpc:~/Desktop/Merge PDF files$
```

Good, now our application has the reference to ***sautinsoft.pdf*** package and we can write the code to convert pdf to docx and other formats.

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



The screenshot shows the Visual Studio IDE with the Explorer pane on the left, the Solution Explorer in the middle, and the Code editor on the right. The Code editor displays the contents of Program.cs, which includes a class Sample with a static Main method and a static MergePdf method. The MergePdf method uses SautinSoft.Pdf to merge three PDF files into a single PDF. The terminal pane at the bottom shows the output of the dotnet restore command, indicating that the projects are up-to-date for restore.

```
8 {
9     0 references
10    class Sample
11    {
12        0 references
13        static void Main(string[] args)
14        {
15            MergePdf();
16        }
17
18        1 reference
19        static void MergePdf()
20        {
21            //SautinSoft.Pdf.PdfDocument.SetLicense("1234567890");
22            string[] inpFiles = new string[] {
23                Path.GetFullPath(@"home/developer/Downloads/MergeFile01.pdf"),
24                Path.GetFullPath(@"home/developer/Downloads/MergeFile02.pdf"),
25                Path.GetFullPath(@"home/developer/Downloads/MergeFile03.pdf")};
26
27            string outFile = @"home/developer/Downloads/Merged.pdf";
28
29            using (var pdf = new PdfDocument())
30            {
31                // Merge multiple PDF files into single PDF.
32                foreach (var inFile in inpFiles)
33                {
34                    using (var source = PdfDocument.Load(inFile))
35                    {
36                        foreach (var page in source.Pages)
37                        {
38                            pdf.Pages.AddClone(page);
39                        }
40                    }
41                }
42                pdf.Save(outFile);
43            }
44            // Show the result PDF document.
45        }
46    }
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS POLYGLOT NOTEBOOK

The template "Console App" was created successfully.

Processing post-creation actions...

Restoring /home/developer/Desktop/Merge PDF files/Merge PDF files.csproj:

Determining projects to restore...

Restored /home/developer/Desktop/Merge PDF files/Merge PDF files.csproj (in 69 ms).

Restore succeeded.

• developer@defaultpc:~/Desktop/Merge PDF files\$ dotnet restore

Determining projects to restore...

All projects are up-to-date for restore.

• developer@defaultpc:~/Desktop/Merge PDF files\$

The code:

```
using System;
using System.IO;
using SautinSoft;
using SautinSoft.Pdf;
using SautinSoft.Pdf.Content;
```

```
namespace Sample
{
    class Sample
    {
        static void Main(string[] args)
```

```

{
    MergePdf();
}

static void MergePdf()
{
    //SautinSoft.Pdf.PdfDocument.SetLicense("1234567890");
    string[] inpFiles = new string[] {
        Path.GetFullPath(@"home/developer/Downloads/MergeFile01.pdf"),
        Path.GetFullPath(@"home/developer/Downloads/MergeFile02.pdf"),
        Path.GetFullPath(@"home/developer/Downloads/MergeFile03.pdf")};

    string outFile = @"home/developer/Downloads/Merged.pdf";

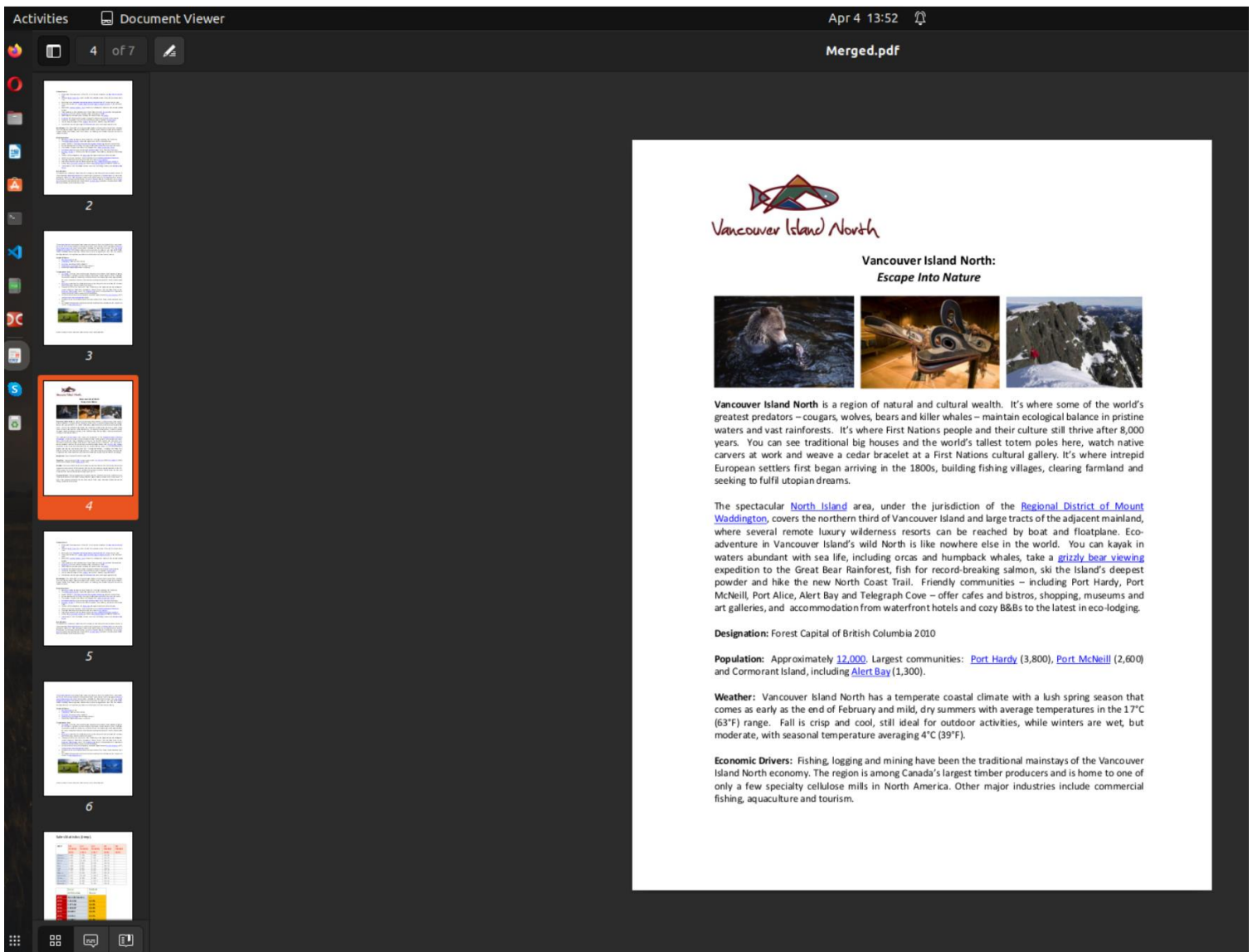
    using (var pdf = new PdfDocument())
    {
        // Merge multiple PDF files into single PDF.
        foreach (var inFile in inpFiles)
        {
            using (var source = PdfDocument.Load(inFile))
            {
                foreach (var page in source.Pages)
                {
                    pdf.Pages.AddClone(page)
                }
            }
        }
        pdf.Save(outFile);
    }
    // Show the result PDF document.
    System.Diagnostics.Process.Start(new System.Diagnostics.ProcessStartInfo(outFile)
    { UseShellExecute = true });
}
}

```

To make tests, we need an input PDF's documents (Merge it).

	MergeFile01	pdf	455.7 K
	MergeFile02	pdf	455.7 K
	MergeFile03	pdf	65.4 K

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



Launch our application and merge PDF files, type the command: ***dotnet run***

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
jorgen@jorgen-linux:~/Desktop/pdf to docx$ dotnet run
Converting successfully!
jorgen@jorgen-linux:~/Desktop/pdf to docx$
```

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

The new file "Merged.pdf" has to appear on the Desktop:

	Merged	pdf	951.3 K
	MergeFile01	pdf	455.7 K
	MergeFile02	pdf	455.7 K
	MergeFile03	pdf	65.4 K

Well done! You have created the "Merge PDF files" 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!