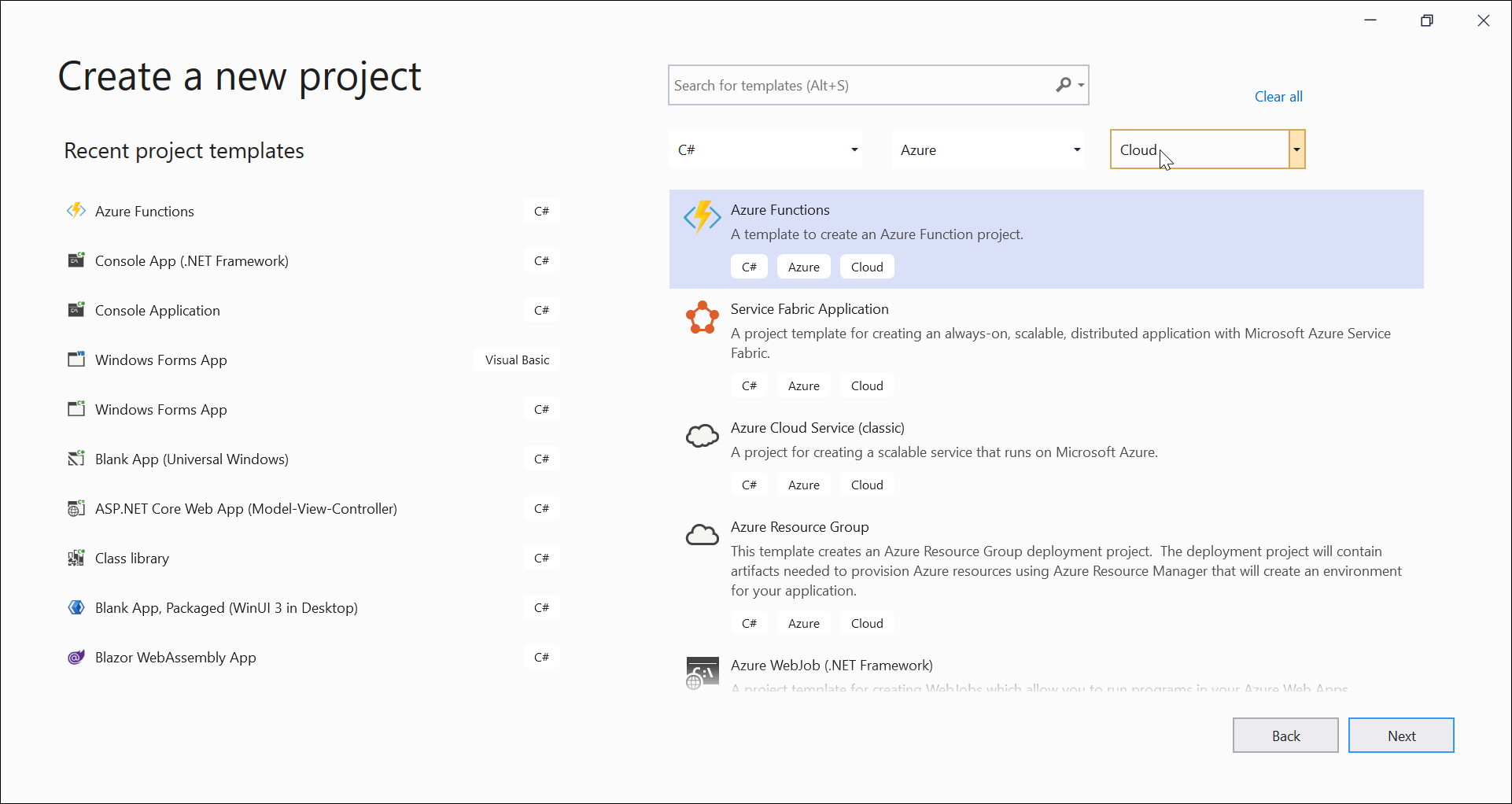
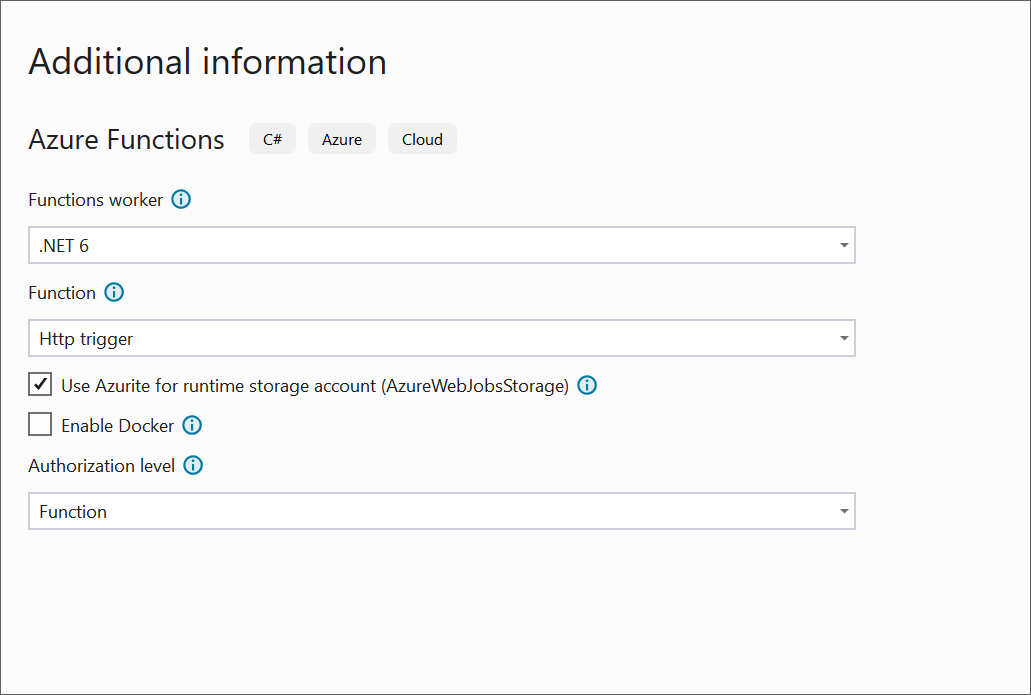
**Convert HTML to PDF in Azure Functions 4.0 in ASP.NET Core**

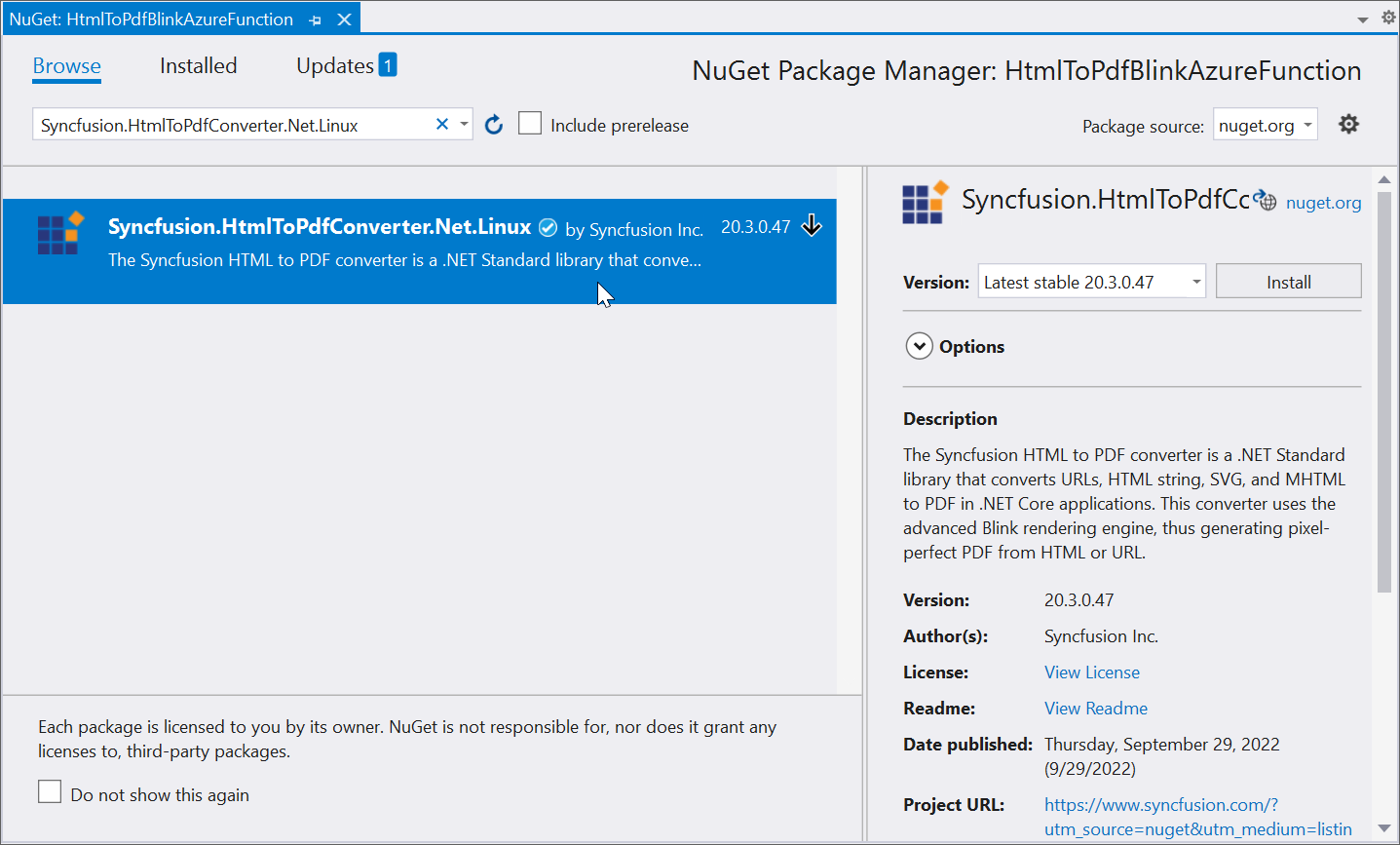
Syncfusion® [**HTML to PDF converter**](https://www.syncfusion.com/document-processing/pdf-framework/net/html-to-pdf) for [**ASP.NET Core PDF**](https://www.syncfusion.com/document-processing/pdf-framework/net) used to convert web pages, and HTML to PDF. Using this library, you can convert any HTML strings or URL or web pages to PDF in Azure Functions 4.0(NET 6).

In this tutorial, we will illustrate the conversion of HTML to PDF using C# with advanced WebKit rendering engine. In addition, our HTML to PDF converter will work seamlessly in various platforms like Azure cloud or web apps, Amazon Web Service (AWS), Docker, WinForms, WPF, ASP.NET MVC, ASP.NET Core with Windows, Linux, and MacOS.

Steps to convert HTML to PDF in Azure Functions 4.0

1. Create an Azure Functions projects. 

1. Select framework to Azure Functions V4 (.NET 6) and select HTTP triggers as follows. 

1. Install the [Syncfusion.HtmlToPdfConverter.Net.Linux](https://www.nuget.org/packages/Syncfusion.HtmlToPdfConverter.Net.Linux/" \t "_blank) NuGet package as a reference to the project. 
2. Include the following namespace in Functions1.cs file to convert the HTML to PDF using C#.
3. **using** Syncfusion.HtmlConverter;
4. **using** Syncfusion.Pdf;
5. **using** System.Runtime.InteropServices;

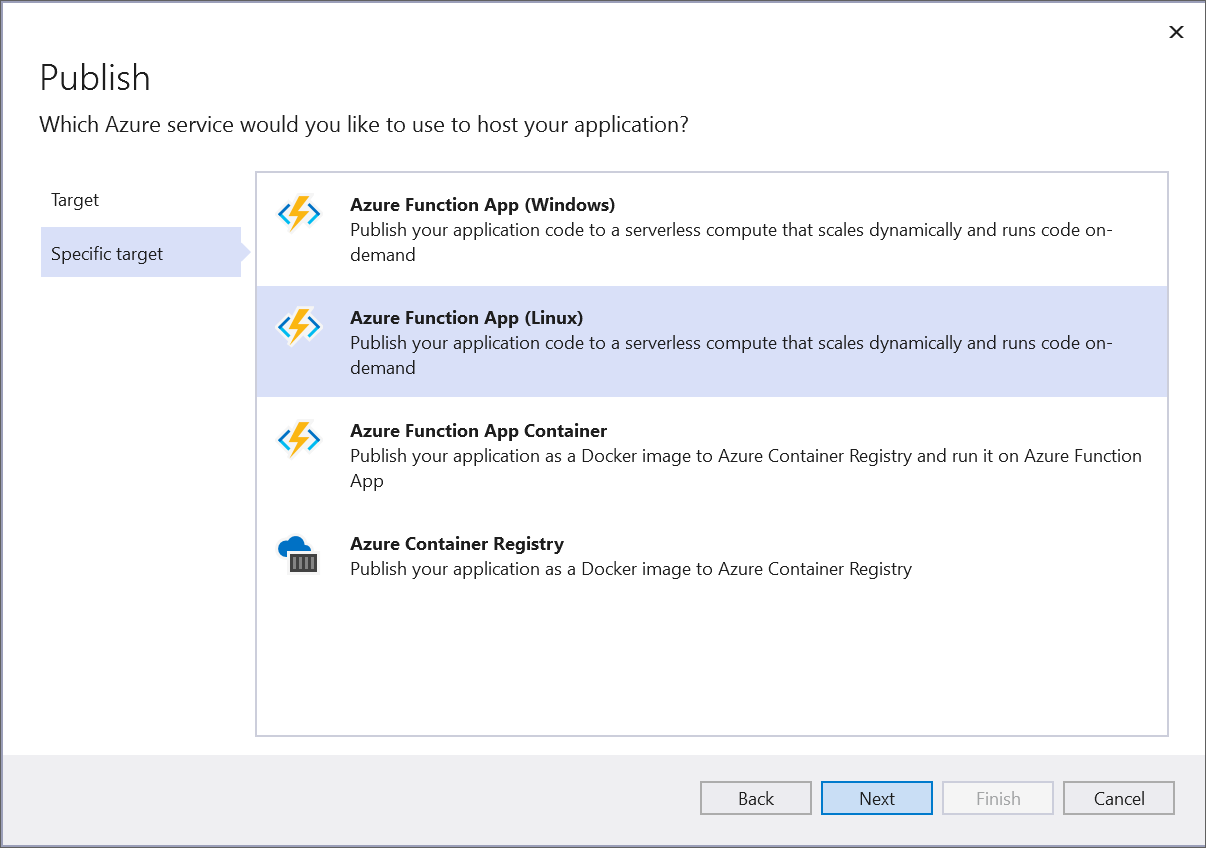
**C#**

1. Add the following code snippet in Function1 class to convert HTML to PDF in Azure Functions V4.
2. [FunctionName("Function1")]
3. public static async Task<IActionResult> Run([HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req, ILogger log, ExecutionContext executionContext)
4. {
5. string blinkBinariesPath = string.Empty;
6. **try**
7. {
8. blinkBinariesPath = SetupBlinkBinaries(executionContext);
9. }
10. catch
11. {
12. throw **new** Exception("BlinkBinaries initialization failed");
13. }
15. string url = req.Query["url"];
17. //Initialize the HTML to PDF converter with the Blink rendering engine.
18. HtmlToPdfConverter htmlConverter = **new** HtmlToPdfConverter(HtmlRenderingEngine.Blink);
19. BlinkConverterSettings settings = **new** BlinkConverterSettings();
21. //Set command line arguments to run without sandbox.
22. settings.CommandLineArguments.Add("--no-sandbox");
23. settings.CommandLineArguments.Add("--disable-setuid-sandbox");
25. settings.BlinkPath = blinkBinariesPath;
27. //Assign Blink Converter settings to the HTML converter
28. htmlConverter.ConverterSettings = settings;
30. //Convert URL to PDF
31. PdfDocument document = htmlConverter.Convert(url);
33. MemoryStream ms = **new** MemoryStream();
35. //Save and close the PDF document
36. document.Save(ms);
37. document.Close();
39. ms.Position = 0;
41. return **new** FileStreamResult(ms, "application/pdf");
42. }
43. Add the following helper methods to copy and set permission to the BlinkBinariesLinux folder.
44. **private** static string SetupBlinkBinaries(ExecutionContext executionContext)
45. {
46. string blinkAppDir = Path.Combine(executionContext.FunctionAppDirectory, "BlinkBinariesLinux");
47. string tempBlinkDir = Path.GetTempPath();
48. string chromePath = Path.Combine(tempBlinkDir, "chrome");
50. **if** (!File.Exists(chromePath))
51. {
53. CopyFilesRecursively(blinkAppDir, tempBlinkDir);
55. SetExecutablePermission(tempBlinkDir);
56. }
57. return tempBlinkDir;
58. }

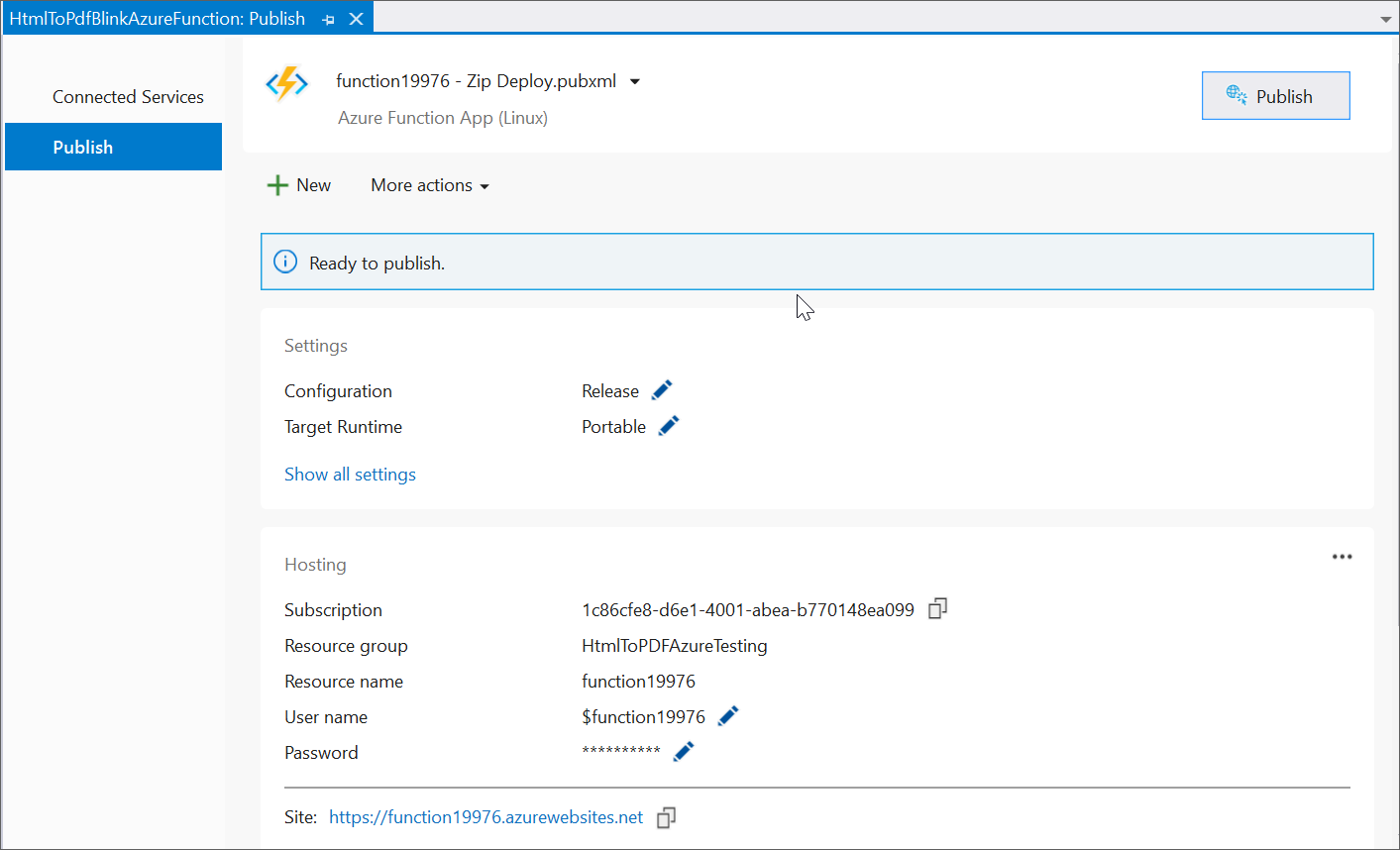
61. **private** static void CopyFilesRecursively(string sourcePath, string targetPath)
62. {
63. //Create all the directories from the source to the destination path.
64. foreach (string dirPath **in** Directory.GetDirectories(sourcePath, "\*", SearchOption.AllDirectories))
65. {
66. Directory.CreateDirectory(dirPath.Replace(sourcePath, targetPath));
67. }
69. //Copy all the files from the source path to the destination path.
70. foreach (string newPath **in** Directory.GetFiles(sourcePath, "\*.\*", SearchOption.AllDirectories))
71. {
72. File.Copy(newPath, newPath.Replace(sourcePath, targetPath), true);
73. }
74. }
76. [DllImport("libc", SetLastError = true, EntryPoint = "chmod")]
77. internal static extern int Chmod(string path, FileAccessPermissions mode);
79. **private** static void SetExecutablePermission(string tempBlinkDir)
80. {
81. FileAccessPermissions ExecutableFilePermissions = FileAccessPermissions.UserRead | FileAccessPermissions.UserWrite | FileAccessPermissions.UserExecute |
82. FileAccessPermissions.GroupRead | FileAccessPermissions.GroupExecute | FileAccessPermissions.OtherRead | FileAccessPermissions.OtherExecute;
84. string[] executableFiles = **new** string[] { "chrome", "chrome\_sandbox" };
86. foreach (string executable **in** executableFiles)
87. {
88. var execPath = Path.Combine(tempBlinkDir, executable);
90. **if** (File.Exists(execPath))
91. {
92. var code = Function1.Chmod(execPath, ExecutableFilePermissions);
93. **if** (code != 0)
94. {
95. throw **new** Exception("Chmod operation failed");
96. }
97. }
98. }
99. }
100. Include the below enum in the Function1.cs file.
101. [Flags]
102. internal enum FileAccessPermissions : uint
103. {
104. OtherExecute = 1,
105. OtherWrite = 2,
106. OtherRead = 4,
108. GroupExecute = 8,
109. GroupWrite = 16,
110. GroupRead = 32,
112. UserExecute = 64,
113. UserWrite = 128,
114. UserRead = 256
115. }

**Public to Azure Functions Linux:**

1. Right-click the project and select **Publish**. Then, create a new profile in the Publish Window. The Blink rendering engine will work in consumption plan. So, you can create the Azure Function App service with a consumption plan.

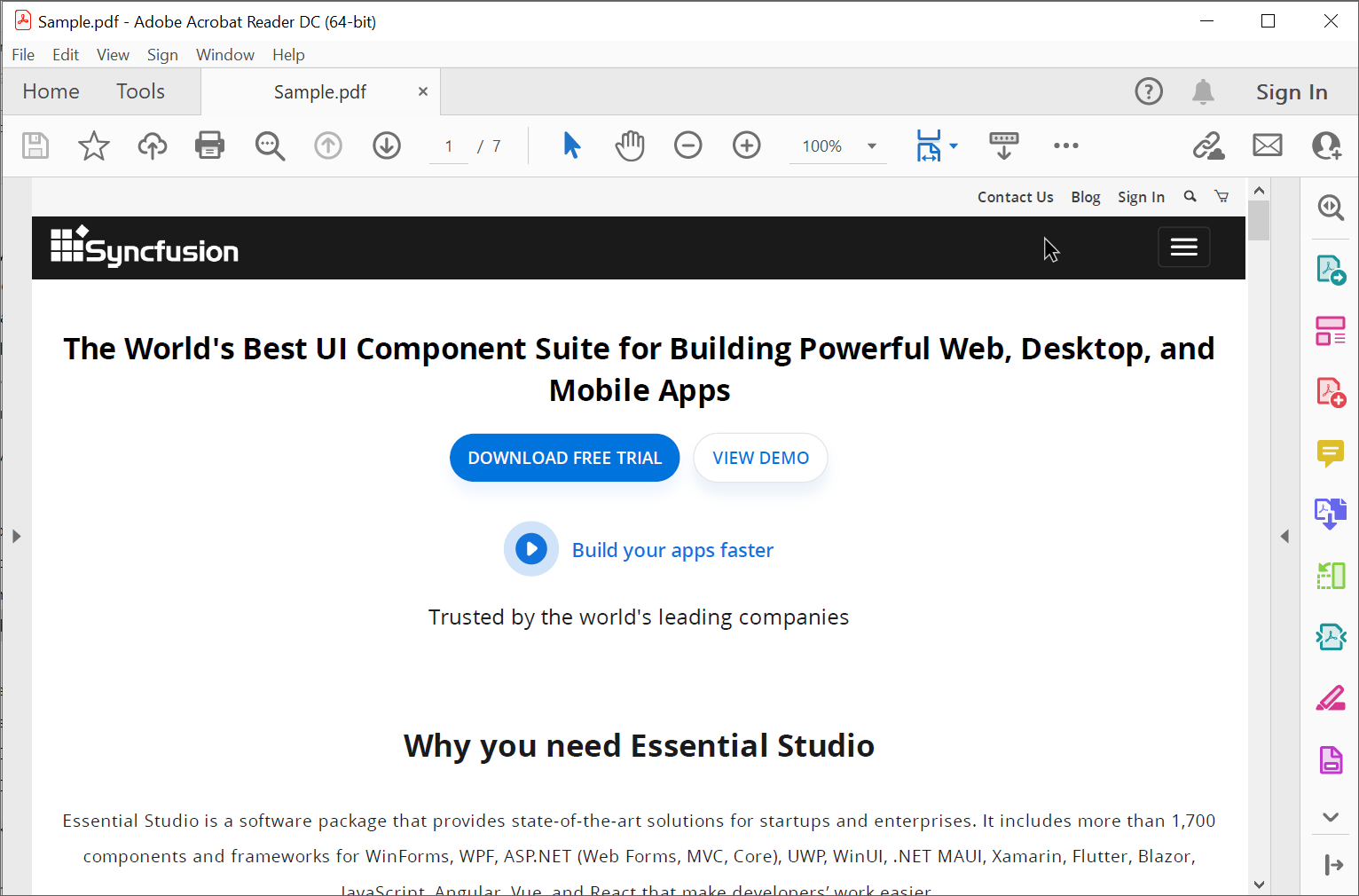


1. After creating the profile, click the Publish button.



1. Now, go to the Azure portal and select the App Services. After running the service, click **Get function URL > Copy**. Include the URL as a query string in the URL. Then, paste it into the new browser tab. You will get the PDF document as follows.

**Example:** https://htmltopdfazurefunctionblink.azurewebsites.net/api/Function1?code=aYCeymbJeHMGIwEEIuCIcNe….​&url=https://www.syncfusion.com



A complete working sample can be downloaded from [HtmlToPdf\_AzureFunctions4.0.zip](https://www.syncfusion.com/downloads/support/directtrac/general/ze/HTMLToPDF_AzureFunctions4.01738776014).

Take a moment to peruse the [documentation](https://help.syncfusion.com/file-formats/pdf/converting-html-to-pdf?_ga=2.177406373.1281575720.1636574891-1699463946.1628662123), where you can find converting HTML pages to PDF document along with respective customization options and features.

Click [here](https://www.syncfusion.com/document-processing/pdf-framework/net/html-to-pdf) to explore the rich set of Syncfusion Essential® PDF features.

An online sample link to [convert HTML to PDF](https://ej2.syncfusion.com/aspnetcore/PDF/HtmltoPDF#/bootstrap5).

**Conclusion**

I hope you enjoyed learning about how to convert HTML to PDF in Azure Functions 4.0.

You can refer to our [**ASP.NET Core PDF's**](https://www.syncfusion.com/document-processing/pdf-framework/net) feature tour page to know about its other groundbreaking feature representations and [**documentation**](https://help.syncfusion.com/aspnet-core/pdf/getting-started), and how to quickly get started for configuration specifications. You can also explore our [**ASP.NET PDF example**](https://ej2.syncfusion.com/net/PDF/Default?#/bootstrap5) to understand how to create and manipulate data.

For current customers, you can check out our [**Document Processing Libraries**](https://www.syncfusion.com/document-processing-libraries) from the [**License and Downloads**](https://www.syncfusion.com/account/downloads?) page. If you are new to Syncfusion®, you can try our 30-day free trial to check out our controls.

If you have any queries or require clarifications, please let us know in the comments section below. You can also contact us through our [**support forums**](https://www.syncfusion.com/forums?), [**Direct-Trac**](https://support.syncfusion.com/create), or [**feedback**](https://www.syncfusion.com/feedback) portal. We are always happy to assist you!