**Task 1: Understanding .NET Framework & .NET Core**

* Gain an understanding of the .NET ecosystem, including the differences between .NET Framework, .NET Core, and .NET 6/7/8.

**Objectives**

* Learn the architecture and benefits of .NET Core and .NET Framework.
* Understand when to use .NET Core vs. .NET Framework.
* Set up the development environment with .NET SDK and Visual Studio.
* Explore NuGet package management for dependency handling

**Architecture:**

* **.NET Core:**

1. .Net Core was introduced in 2016 by Microsoft which is a lightweight, cross platform, and high performance version.
2. Provides support for modern applications that require microservices, cloud,etc.
3. It has a very modular design, which allows developers to keep only the components that are necessary, thus making it lightweight and efficient.

* **.NET Framework:**

1. .NET Framework (used since 2002) was based on Common Language Runtime(CLR).
2. It had support for windows-only applications. This framework was not cross platform.
3. Supported Desktop Applications(WPF, WinForms) and ASP.NET(MVC, Web Forms)

**Benefits:**

* **.NET Core:**

Cross-platform compatibility is a major benefit of.NET Core; apps built with it can operate on Windows, macOS, and Linux, among other operating systems. Its the ideal option for creating cloud-based and microservices architectures   
  
Because of its extensive community support and regular updates,.NET Core remains at the forefront of technology, giving developers access to the newest tools and improvements.

* **.NET Framework:**

The.NET Framework is a great option for creating reliable and secure applications because of its strong security features, such as code access security, and its vast ecosystem, which supports a wide range of third-party libraries and tools.   
  
Its versatility is increased by support for Windows-specific technologies such as Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF), and Windows Universal Apps, which enable developers to create extensive Windows-based apps.

**When To Use:**

* **.NET Core:**

1. When you are developing an application that can run on either Windows,Linux or Mac, dotnet core is the way to go as it is cross platform.
2. Use dotnet core for scalable solutions.
3. DoteNet core is used when you want to develop microservices, cloud applicationsand modern web services.

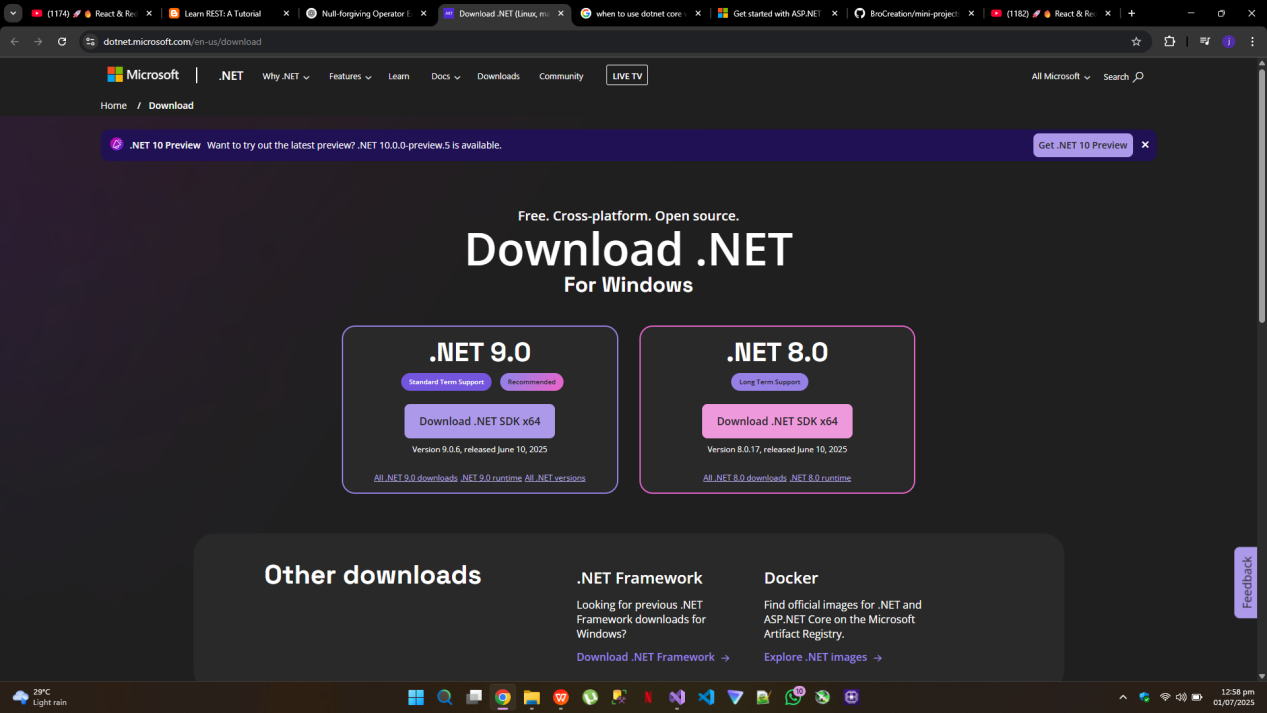
* **.NET Framework:**

1. If you have applications that are built on dotnet framework and are functioning well, then there is no need to migrate to dotnet core.
2. Because of its strong support for Windows Forms and WPF applications, dotnet framework may be better suited for Windows based desktop applications.
3. When you require functionality that is only available in dotnet framework, it is the way to go.

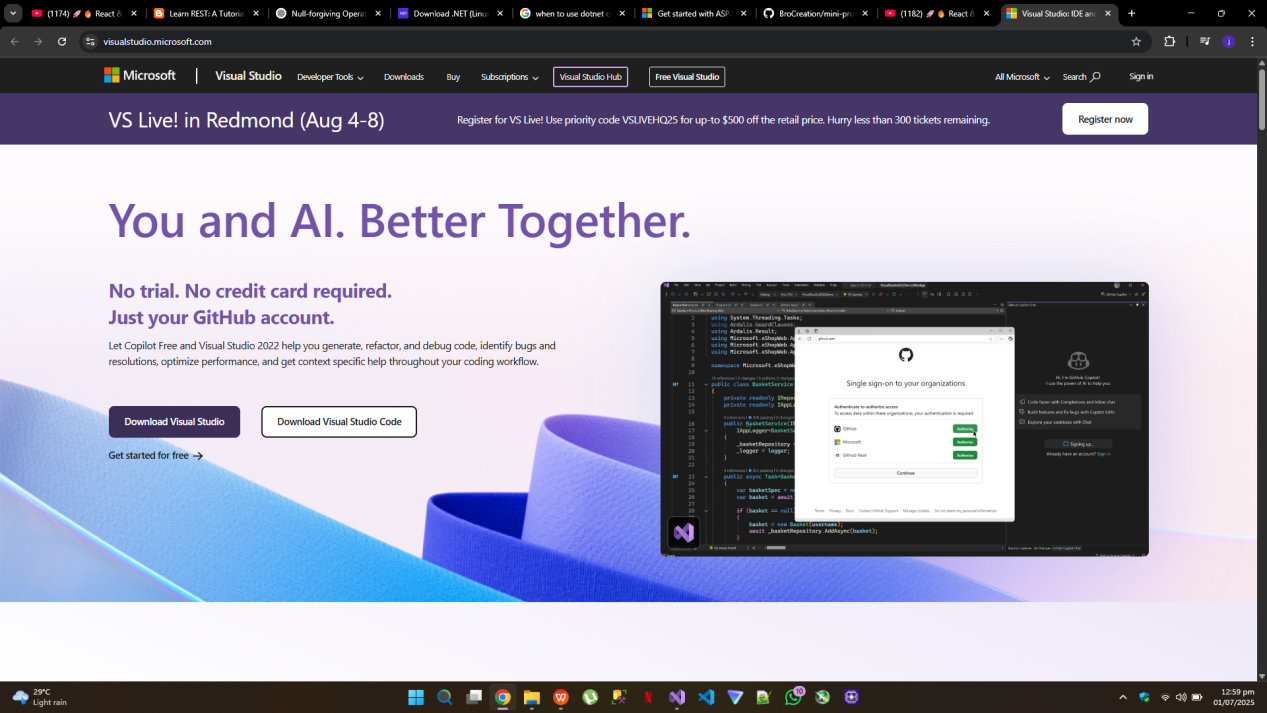
**Set up the development environment with .NET SDK and Visual Studio:**

**Download and Install:**

* DotNet SDK from : <https://dotnet.microsoft.com/en-us/download>:



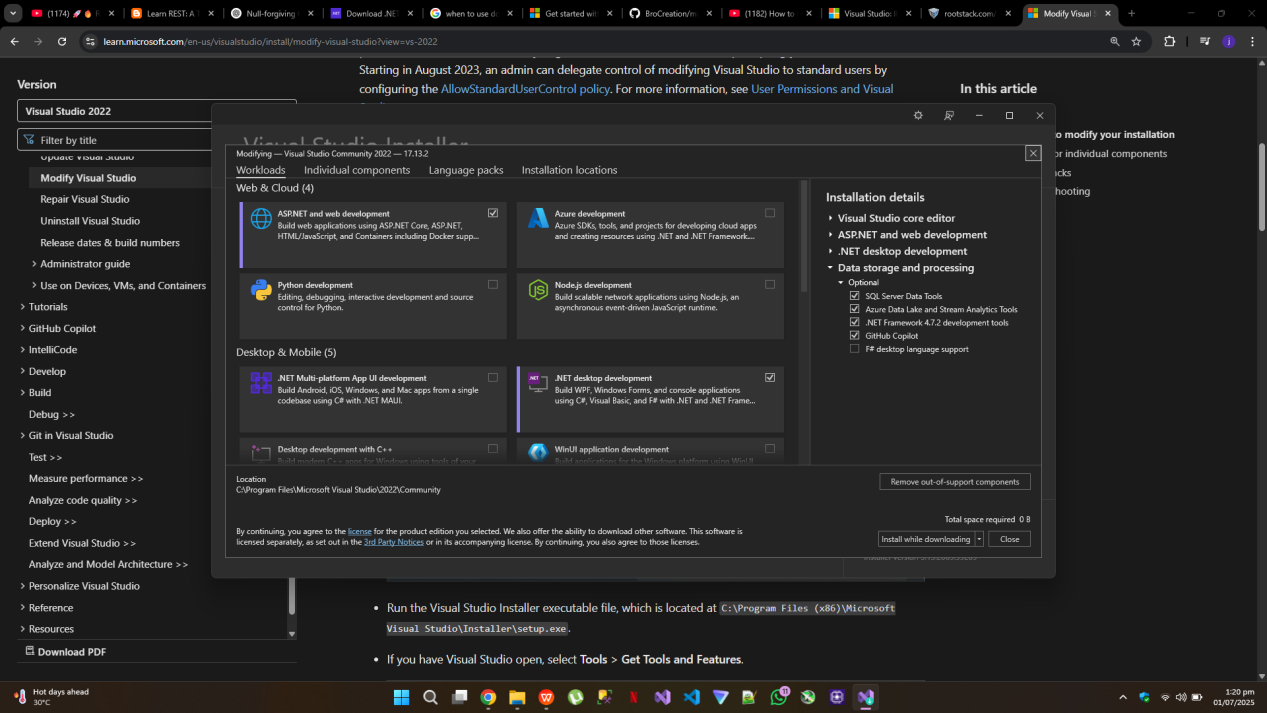
* Microsoft Visual Studio(Community Edition as it is free) from : <https://visualstudio.microsoft.com/>:



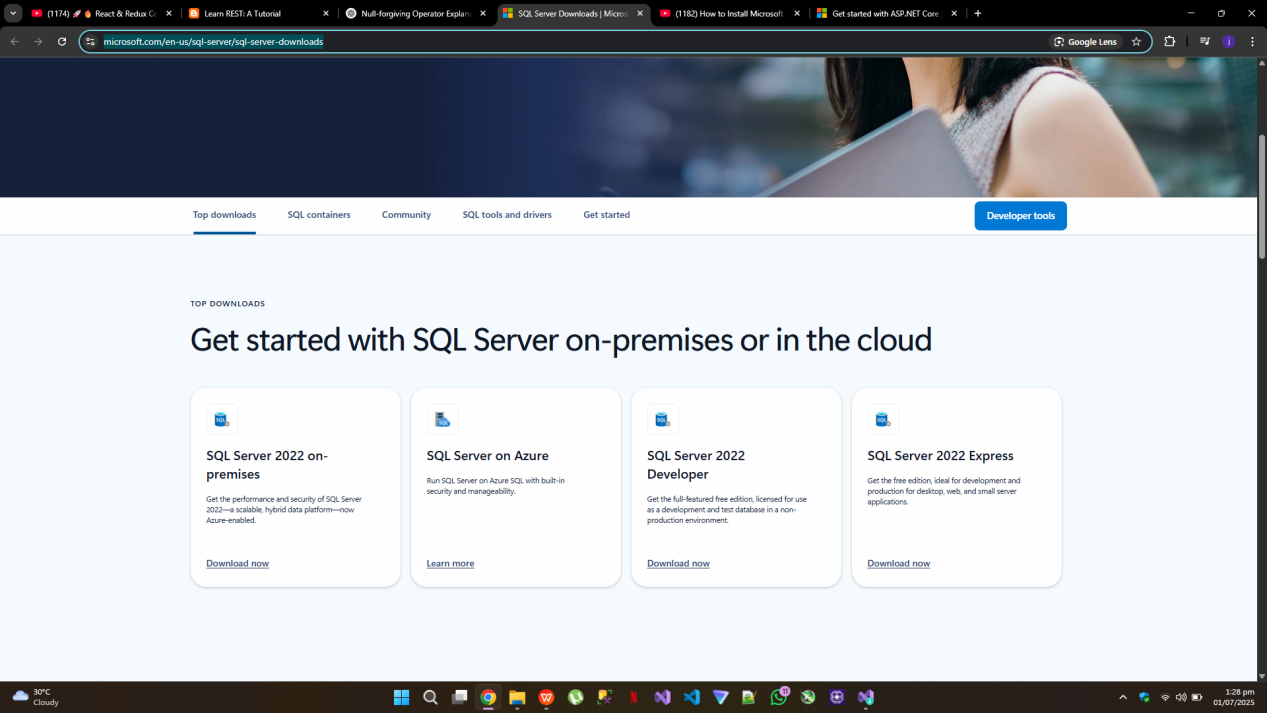
Make sure to:

Install the **ASP.NET and web development,Data storage and processing** workload.

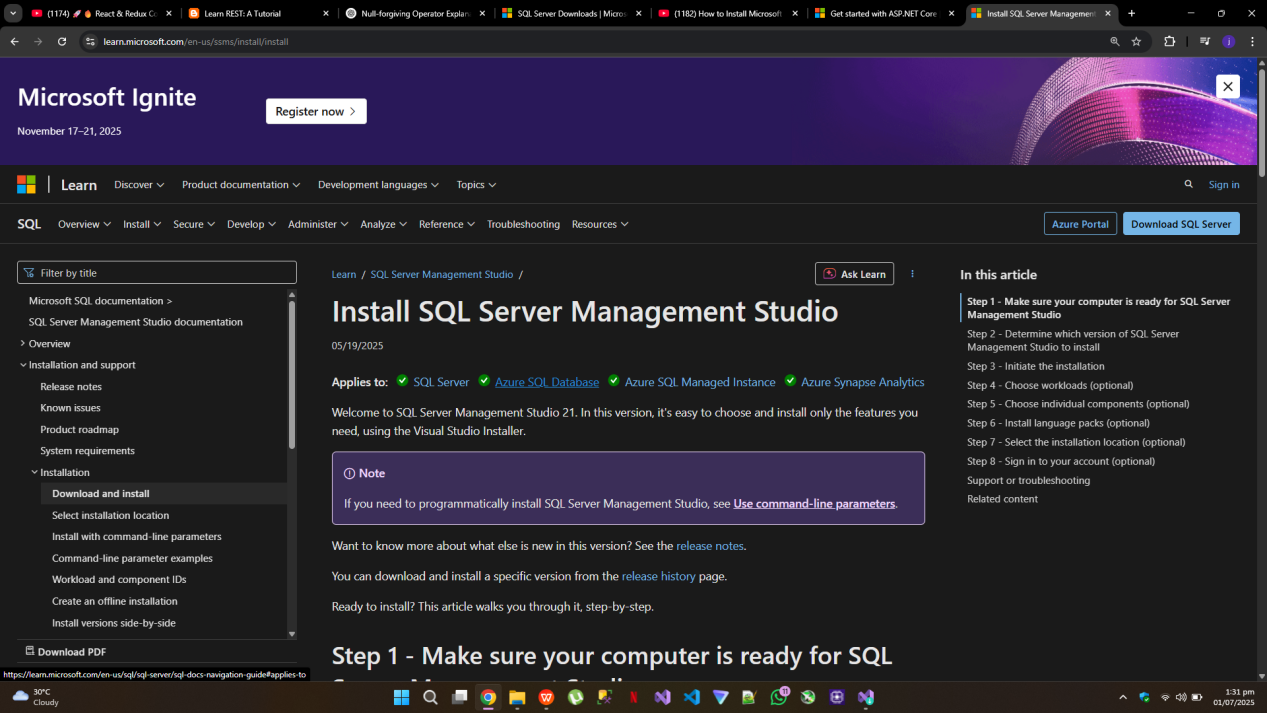
Install **.NET Desktop Development** for using WinForms or WPF, and console application using C#.



* SQL Server(Express Edition) from : <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>

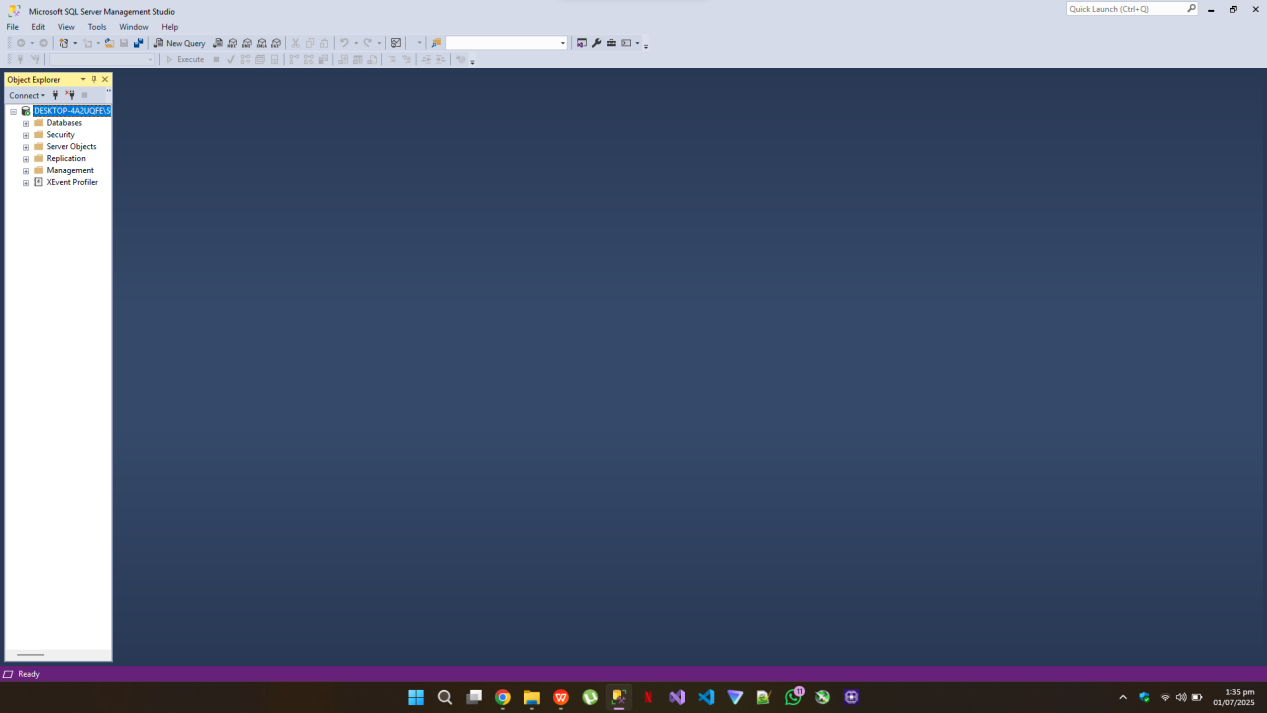


Keep the installation type to basic and keep the default settings and install. After installation is complete, click **Install SSMS** to install SQL Server Management Studio.it will redirect to following page:



SSMS allows to:

* Browse databases
* Run SQL queries
* Inspect tables, views, and stored procedures



Now, your development environment is setup and you can connect your dotnet core projects with sql server as well using the nuget packages and appsettings.json file

**Explore NuGet Package Management:**

1. **Nuget is the package manager of Dot Net (like npm of Node.js)**
2. **Use it to install libraries(Entity Framework, Stripe, etc.)**
3. In Visual Studio:

* Right-click your project > **Manage NuGet Packages**
* Click on Browse Tab
* Search for a package, e.g., Stripe
* Click Install

