# Introduction to .Net

* It is a free, open-source development platform for a variety of applications (games, website, console, etc.)
* Essentially, it is a collection of languages and libraries that can work together to build many different types of applications.

## Different types of open-source developer platform

* .Net 5.0
  + The one you have right now
  + We are using this since it has the most support for different types of applications compared to the other two
    - It is because it is the successor of both .Net core and .Net framework
* .Net Core
  + Essentially it is the older version of 5.0
  + So less support
* .Net Framework
  + It is usually used to create windows-only desktop or server based applications

# What is C#

* It is an object oriented and type-safe programming language
  + Object oriented just means that everything is based on objects and the relationships between them
* Type-safe just means once you set a type you can’t change it.
  + So whenever you put a string in an integer field, you’ll get an error/exception

# Introduction to Git

* It is a Version Control System
  + It allows you to manage the changes/development made in a project (Mostly a coding project)
  + More specifically used **Distributed Version Control System** (DVCS)
* It records any changes made to a project and maintain a history tree that contains the state of the project at a certain point.
  + Laymen’s term, it tracks any changes you’ve made to your files and records them.

## Pros

* It allows you to backtrack to previous version of your code if your current version is too unstable.
* It is incredibly helpful when working on a team environment.
* If local files get corrupted, you can just get the files stored in the cloud to restore your project.

## Cons

* You might have merge conflicts.
  + Scenario which one person worked with the same file as another person and git does not know which modified version to use/store to the main repo.
* Quite confusing when working with it the first time.

## Repository

* This is where your code is stored.
* There is a local repository (Your computer) and a remote repository (Your github repository) that is hosted on a web.
* **NOTE:** Git and Github are two different things
  + GIT – DVCS
  + Github – just a remote repository that you can upload your local repository to update it.

## CLI command

* Git init – Creates an empty local git repository in the folder.
* Git status – Check which files are in staging and which aren’t.
* Git add . – Will add every file to staging (EXCEPT for ignored files in .gitignore)
* Git commit -m ‘Some comment’ – Records the changes to the local repository.
  + -m will let us attach a message/comment to that commit
* Git branch – Will create, delete, or list branches for us.
  + -M will move us to that branch
* Git push -u [remote link you created] [branch] – push the changes you made to a remote repository.
* Git pull – will pull the changes in the remote repository to your local repository.

# Application Architecture

## Separation of Concern

* It is a concept of organizing our code.
* We want our code to follow a certain theme or all the code are there to do a certain functionality.
* We can do this by leveraging classes and other grouping mechanism to group data and logic together.
* This is a first but **important** step to writing readable, extendable, and maintainable code.

## Classes

* They are the building blocks for your code.
* They are essentially the blueprints to create objects that you process in your programs.
* Besides being used to structure your data, you also use classes to encapsulate logic and data together.

## Namespace

* Logical grouping of types that follow a certain theme of functionality.
* To utilize the classes located in a different namespace, use the *using* keyword.
* It is the same as Java packages.

## Projects

* They contain all the files that are compiled into an executable, library, or website.
* They can also contain compiler settings or some configuration files that are needed to customize your components or services.

## Solution

* They are simply containers for one ore more related projects, along with build information.
* They are the final packaging of your application.