# GIT PowerShell commands

* Git add .
  + <https://git-scm.com/docs/git-add>
  + Command updates the index using the current content found in the working tree, to prepare the content staged for the next commit
* Git push origin main
  + <https://git-scm.com/docs/git-push>
  + Updates remote refs along with associated objects
* Git commit -m “Comment”
  + <https://git-scm.com/docs/git-commit>
  + Record changes to the repository
* Git checkout -b main
  + <https://git-scm.com/docs/git-checkout>
  + Switch branches or restore working tree files
* Git config
  + <https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup>
  + Used for initial setup of GIT. Allows you set get and set configuration variables
* Git remote add origin <https://github.com/user/something.git>
  + <https://git-scm.com/docs/git-remote>
  + Manage set of tracked repositories whose branches you track
* Git init
  + <https://git-scm.com/docs/git-init>
  + Creates an empty GIT repository or reinitializes an existing one

# Dotnet commands

* Dotnet ef database update
  + <https://learn.microsoft.com/en-us/ef/core/cli/dotnet#dotnet-ef-database-update>
  + Updates the database to the last migration or to a specified migration
* Dotnet ef migrations add “Comment” -o Data/Migrations
  + <https://learn.microsoft.com/en-us/ef/core/cli/dotnet#dotnet-ef-migrations-add>
  + Adds a new migration
* Dotnet ef migrations remove
  + <https://learn.microsoft.com/en-us/ef/core/cli/dotnet#dotnet-ef-migrations-remove>
  + Removes the last migration, rolling back the code changes that were done before the latest migration
* Dotnet build
  + <https://learn.microsoft.com/en-us/dotnet/core/tools/dotnet-build>
  + Builds the project and all its dependencies

# Visual Studio C#

* <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/>
* DbContext
  + <https://learn.microsoft.com/en-us/dotnet/api/system.data.entity.dbcontext?view=entity-framework-6.2.0>
  + A DbContext instance represents a combination of the Unit of Work and Repository patterns such that it can be used to query from a database and group together changes that will then be written back to the store as a unit. DbContext is conceptually similar to ObjectContext
* DbSet
  + <https://learn.microsoft.com/en-us/dotnet/api/system.data.entity.dbset-1?view=entity-framework-6.2.0>
  + A DbSet represents the collection of all entities in the context, or that can be queries from the database, of a given type. DbSet objects are created from a DbContext using the DbContext.Set method
* DbContextOptions
  + <https://learn.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbcontextoptions?view=efcore-6.0>
  + The options to be used by a DbContext
* modelBuilder.Entity
  + <https://learn.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.modelbuilder.entity?view=efcore-6.0>
    - modelBuilder.Entity<CoffeeType>().HasData(coffeeData);
  + This is used in our context to build a database model of CoffeeType with table coffeeData
* Appsettings.json
  + Add in connection string to allow for database connection
* Async Task
  + <https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/>
* [HttpGet]
  + <https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-6.0>

# Angular

* Ng new [projectName]
  + <https://angular.io/cli/new>
  + Creates a new angular workspace
* NG serve
  + <https://angular.io/cli/serve>
  + Builds and serves app