Week 1 - Wednesday

Outline

- GitHub
- .NET Common Language Runtime
- Collection
- QC Orientation
- Working with Types
- Was given a problem on Code Signal (HackerRank/LeetCode)

GitHub

Git Bash - command line environment (ls, cd, mkdir)

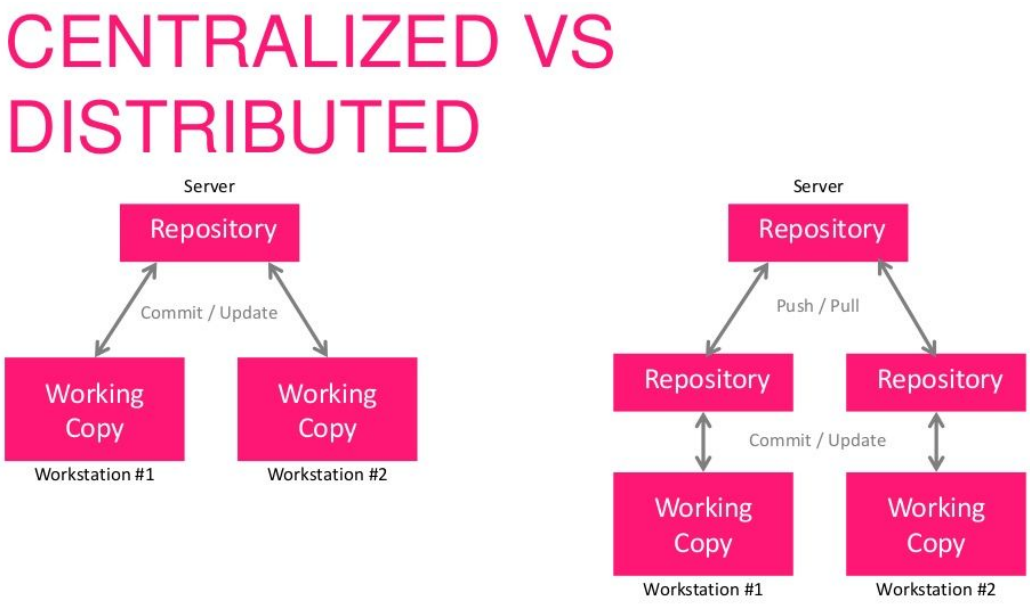
Git - is a version control system (VCS)

- git clone, git pull
- Git is distributed but it can do both

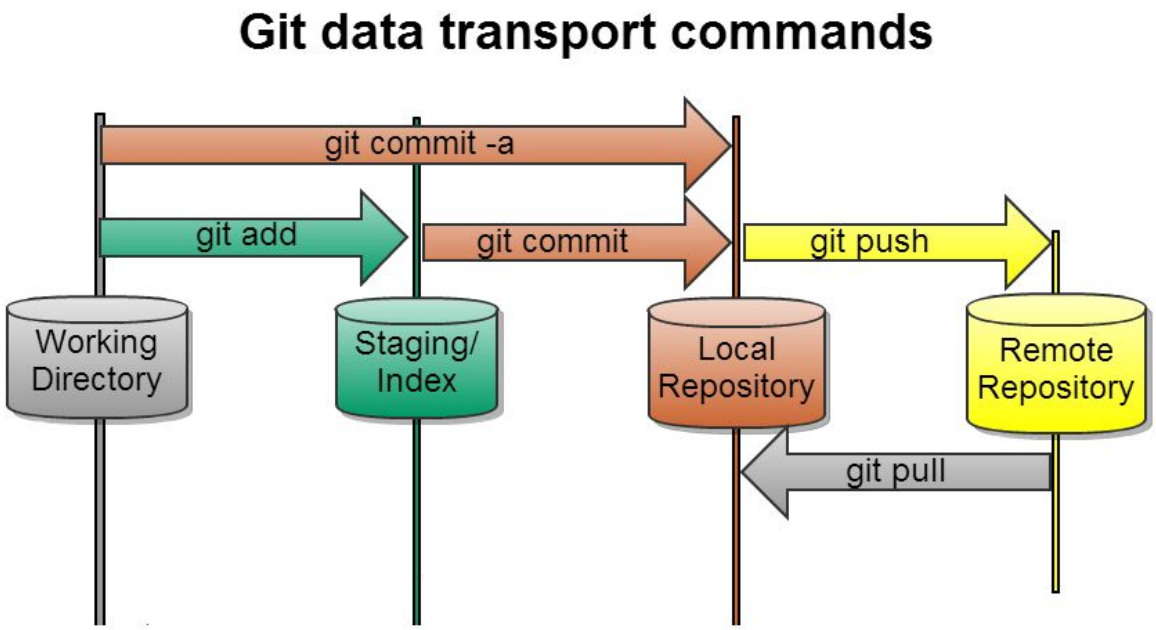
Distributed Version Control System (DVCS)

Central Version Control System

- Subversion (git)
- Mercurial (Hg)
- TFS



Git Workflows



Working Directory - our laptop

- What every program besides Git can currently see

Staging/Index - no one can understand it besides git (temporary)

- We prepare commits here

Local Repository - stores all commits from the projects history (permanent)

- Stores a graph of command

Remote Repository - GitHub (permanent)

- Source of truth

If repository trainer-code/ contains .git/, README.MD, notes.txt
Working Directory contains README.MD and notes.txt
Staging/Index contains .git

git status - displays state of the working directory and Staging/Index
git clone
git diff - compares the working directory and staging index
git diff --cached
git log - show us a picture of our commits

Step-by-step walkthrough
cd/c/revature
git clone https://github.com/1904-apr22-net/kevin-code.git
git status
git add <path>

- We'll normally use: " git add . "
    - " . " in bash means current directory

git commit (if we need the editor)

- Opens "nano editor"
- Each commit should be small enough so that we can summarize what it does in a single sentence. If it longer than that, consider breaking it up into multiple commits.

One time thing
git config <email address>
git config <name>

In Nano
Save is "control + O"
Exit is "control + G"

git commit -m "writing notes"

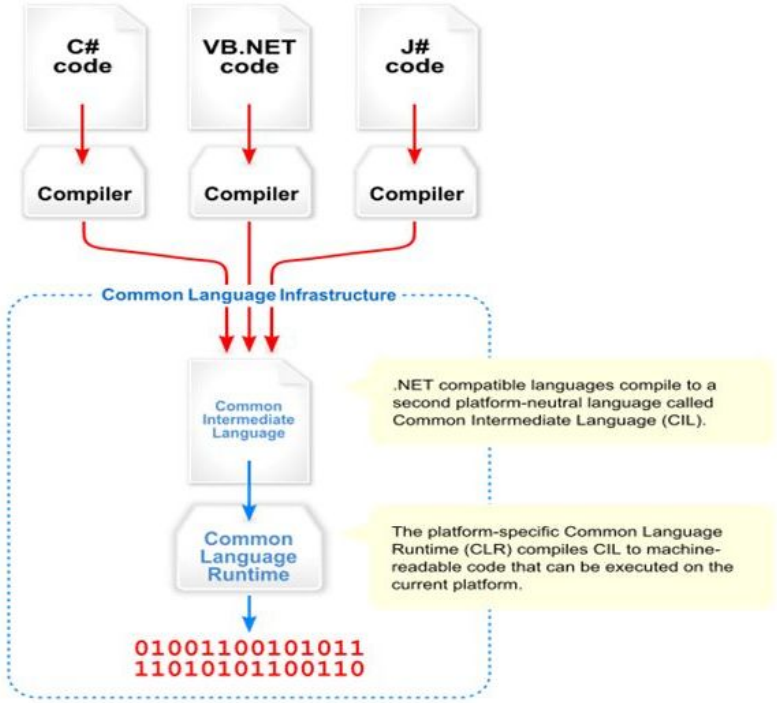
- if we don't need the editor

git push

- Will will push our changes to Git

.NET

- Consists of many different languages (C# is the most popular)
    - C#, F#, VB.NET, C++, Python, Java
    - They are all share a common runtime environment
    - Interoperability across languages



- Compiled by .NET SDK
- **Intermediate Languages** (IL) - are packaged into .dll or .exe files (IL, CIL, MSIL)
    - Works across platform
    - Read by Common Language Runtime (**CLR**)
- **CLR** - translates IL into what the current machine knows what to understand using JIT
    - Just-In-Time (**JIT**) -> native code -> CPU
    - Memory management
    - Base libraries
    - Common Type System (**CTS**)
- Garbage Collector (GC)

.NET Framework

- Windows only
- Version 4.7
- Has more stuff

.NET Core (our focus)

- Cross-platform
- Version 2.2
- Increasingly as much stuff as framework

.NET Standard is the interface whereas .NET Framework/Core are just two implementation of the interface. Some overlap and some difference.

Mono - originally, 3rd party port of Framework to Mac/Linux

Common Language Infrastructure - Any .NET implementation must have

- Virtual execution system (**VES**)
    - Framework: **CLR**
    - Core: CoreCLR
- At least one language Compiler
    - Framework: CSC
    - Core: Roslyn
- Base class library (BCL)

Common Language Runtime

- **BCL** - base class library
    - Simple runtime library for modern programming language

- **CIL** - common intermediate language
    - Lowest level human readable programming language

- **CLI** - common language infrastructure
    - Describes executable code and a runtime environment that allows multiple high-level language

- **CLR** - common language runtime
    - Virtual machine used by microsoft to take CIL code from source code to machine native code

- **CTS** - common type system
    - Responsible for understanding all the data types and converting them into CLR understandable format

- **JIT** - just in time
    - Converts CIL or IL to (machine code)/(native code)

- **VES** - Virtual Execution System
    - Provides an environment for executing managed code
    - Implemented by **CLR**

- **Garbage Collection** (GC) - automatic memory manager

Big picture overview
Source Code -> (Save File) ---- compiles source code into **CIL**/**IL**

CLR provides the environment in which **CIL**/**IL** can run

- **CIL**/**IL** are compiled by **JIT compiler** into native code

Collections

- **Arrays**
    - Use foreach whenever possible
        - Less margin for error when compared to a for loop
    - **Creating 2d Arrays**
        - We can put arrays inside arrays
            - int[][] twoD = new int[6][1]
        - We can create multidimensional aray
            - int[,] twoDMulti = new int[4, 5];
            - twoDMulti[2, 3] = 5;
    - We usually avoid arrays in C# unless there is a performance need

- **ArrayList** - an array of objects whose size is dynamically increased as required.
- **List** - list of objects that can be accessed at index
- **Set**/**Hash** - a collection of key/value pairs that are organized based on the hash code of the key.
    - Duplicates are allowed, but do not count
    - No defined order
- **Dictionary** - a collection of key/value pairs that are organized based on the key.
    - Keys are unique and each key is paired with a one value
- **Stack** - Represents a last in, first out (LIFO) collection of objects.
- **Queue** - Represents a first in, first out (FIFO) collection of objects.

Working with Types

- **Casting** - is a way to convert values from one type to another
    - Don't really use this too often as arrays and array lists are outdated

```
// casting - convert what's on the right to the given type on the left
// will succeed
int num - (int)numList[0];

// will fail
string s = (string)numList[1];
```

- **As** - is a way to convert
    -
- **Generics** - we can write code for many different types and then when we need the code, we'll decide at that time what the type will be.
    - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/collections
    - var salmons = new List<string>();
    -

Quality Control Interview: Dylan McBree

- Will be QCing us week by week
- **Readiness Assessment**
    - Tech screening.
    - Technical and soft skills
- **Weekly Evaluation**
    - Written evaluation (online) and interview
- **Weekly Quality Check**
    - Usually tuesday
    - Technical & Soft skill readiness
    - Content coverage
    - Content delivery
- **Panel Interviews**
    - Expert external interview to ensure comprehensive quality
- **Resume preparation**
    - Best practices and expert feedback
- **Project showcase**
    - Demo full-stack application to experts and stakeholders
- **Certifications**
    - Achieve industry leading certification
- **Final check**
    - Conducted by staging manager (Julie)
- **Ongoing project support**
    - Provide support (projectsupport@revature.com)
    - Feedback to improve training

Things to Review

- Be able to explain CLR Diagram to someone
- Know all of these acronyms:
    - BCL, CIL, CLI, CLR, CTS, JIT, VES
- Practice debugging code on visual studio

Review Activity

- Try creating a “Calculator”
- Try creating a “Bank Application” using what we learned in class

Homework Assignment

- Solve at least one problem on https://app.codesignal.com/