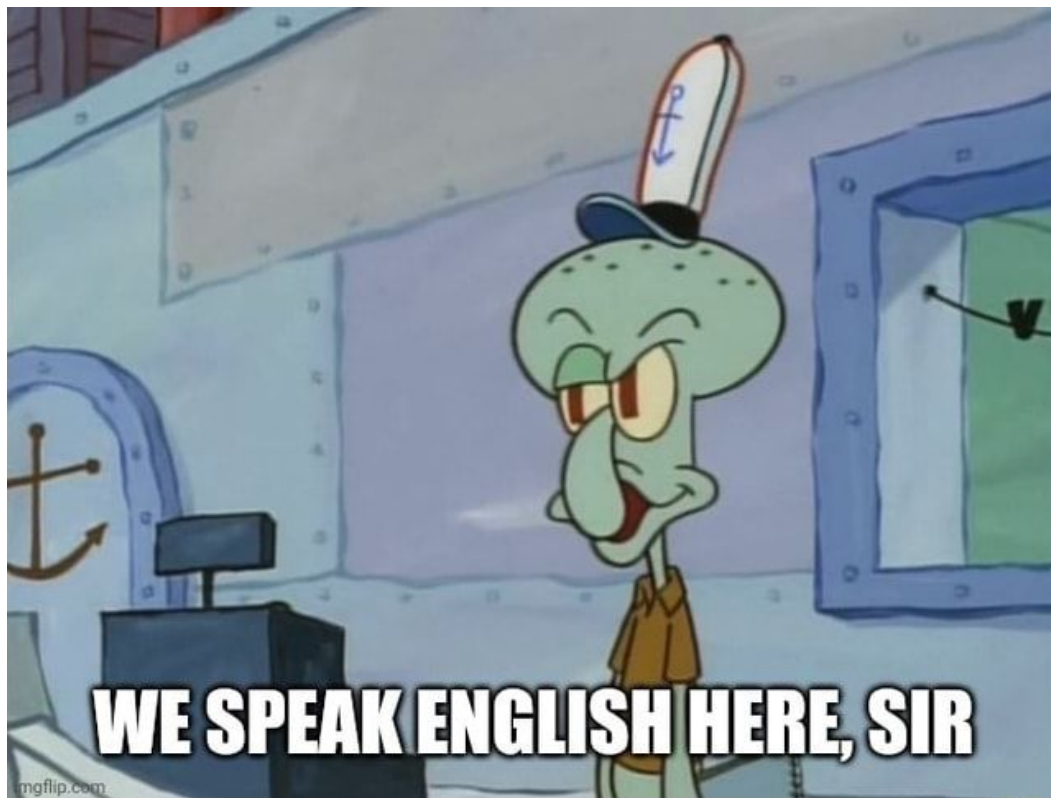




# Software Engineering I

1st lecture





# Agenda

1. Mandatory information about the course.
2. What the lab classes and assignments will look like?
3. Intro to Git.



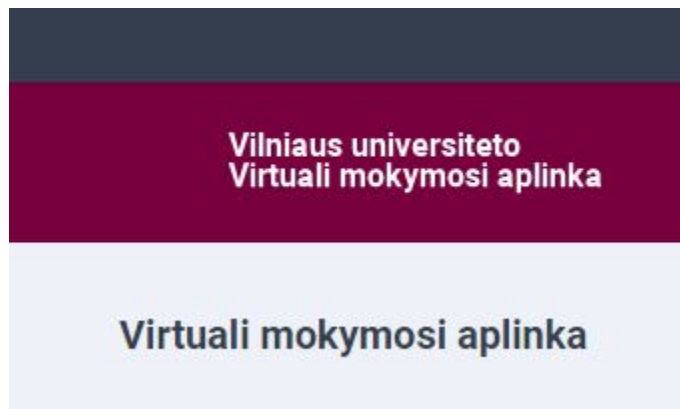
# Contact info

- Email:
  - [tomas.smagurauskas@mif.vu.lt](mailto:tomas.smagurauskas@mif.vu.lt)
- In the subject of the email, provide this:
  - “PSI I: {your topic}”
- More info: Moodle <https://emokymai.vu.lt/>



# Course material

- Slides will be posted in Moodle (VMA) after the lecture.
- All other important material will there as well.





# What the lecture looks like

Lectures are 3 hours long!

- That means 3 academic hours (45 mins x 3)
- That means it is 2 hours 15 minutes
- That means we will most of the times go like this:
  - 16:00 - 17:00 lecture
  - 17:00 - 17:10 break
  - 17:10 - 18:25 lecture

The course is worth 10 ECTS credits.





# Purpose of the course unit

To get acquainted with software development methods using C# programming language and .NET Core framework, to consolidate knowledge of object-oriented programming.



# Learning outcomes of the course unit (1)

- Design, implement and develop applied programs, apply code reviews;
- Apply knowledge of software systems engineering, make qualified design and architectural decisions while expanding the functionality of the developed system;
- Combine theory and practice using .NET framework technologies and developing OO application systems;





## Learning outcomes of the course unit (2)

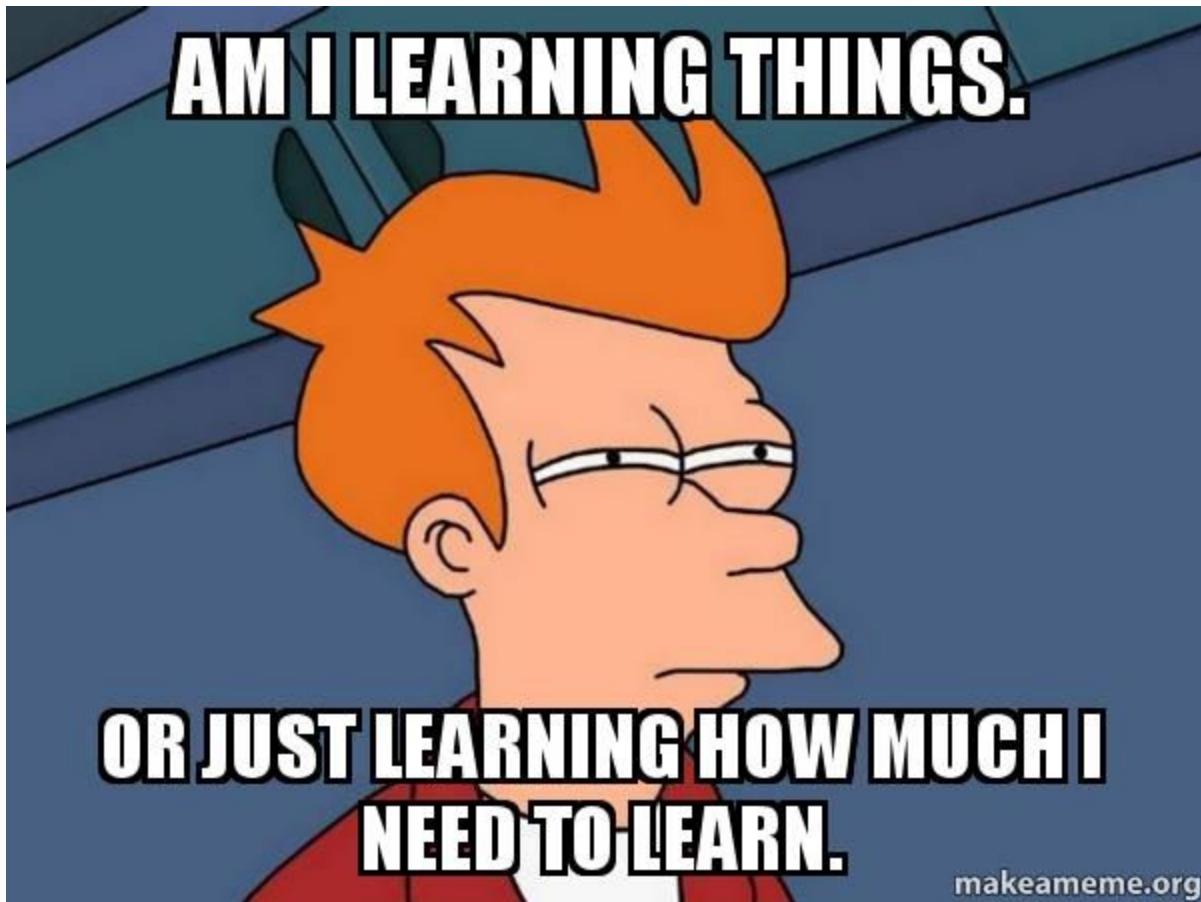
- Develop the knowledge about data types, named and optional arguments as well as other new features of C# programming language;
- Program in C# independently and in a team, applying basic OO design templates using C# programming environment;
- Recognize the need for continuous learning and will have the initial skills;
- Work in the team - on site and remotely;

how long have you  
been programming ?

- 10 Years

So you have lots of  
knowledge in this field.





**AM I LEARNING THINGS.**

**OR JUST LEARNING HOW MUCH I  
NEED TO LEARN.**



# Course overview (1)

Four main sections:

1. Introduction to basic C # constructs;
2. Exceptional C # and OOP features;
3. Working with databases and APIs;
4. .NET and OO solutions and patterns overview;



# Course overview and basics: 1, 2 lecture

- Acquaintance with C# programming language;
- Loops/operators;
- Type systems;
- Application build tools, .NET Core compatibilities with different operating systems;
- Code versioning systems (GIT);
- Code reviews;
- Generic types and methods;



## Course overview: 3, 4 lecture (1)

- Value types;
- Reference types:
  - Classes and their interaction: fields, properties, methods, nested classes;
- Common .NET interfaces:
  - IComparable, IComparer, IEquatable, ICloneable, IEnumerable, IEnumerator;

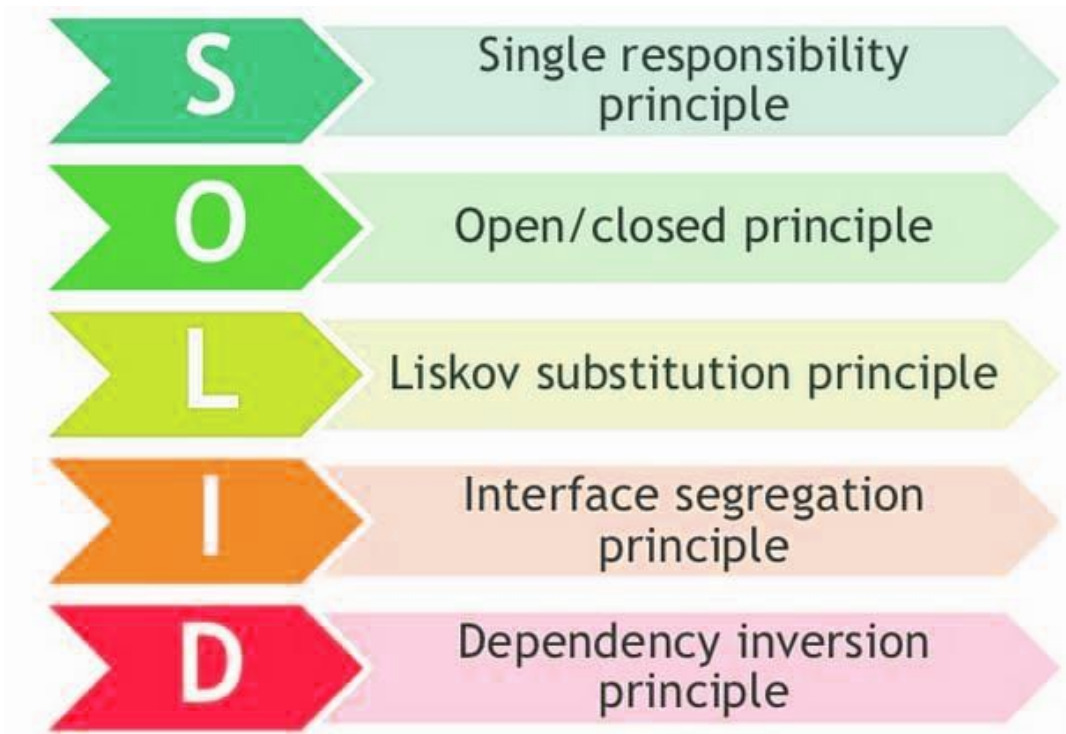


## Course overview: 3, 4 lecture (2)

- Type conversions:
  - Allowed conversion (*Widening, narrowing, implicit, and explicit, checked* keyword),
  - Casting;
  - Conversions of incompatible types;
  - *is* and *as*;
  - *boxing* and *unboxing*;
- Software construction;



# Course overview: 3, 4 lecture (3)







# Course overview: 5, 6 lecture (1)

- Creation of objects:
  - Lazy / object initializers / anonymous type / equals;
- Extension methods;
- Typical OOP mistakes and how to avoid them;
- Object lifecycle;
- String type variables;



# Course overview: 5, 6 lecture (2)

- Software system construction;
- Business needs analysis;
- Software system modification and maintenance;
- Delegates: *Action* and *Func*;
- Anonymous methods;
- *Lambda* expressions;
- Events



# Course overview: 5, 6 lecture (3)

- Expression trees;
- Introduction to LINQ;
- Working with data;
- Collections;



# Course overview: 7, 8 lecture

- Working with databases;
- Introduction to ORMs;
- Entity Framework Core;
- Serialization;
- Attributes;
- Exceptions and their handling;
- Introduction to project management;
- Basics of Agile;



# Course overview: 9,10 lecture

- Web services:
  - SOAP
  - GraphQL
  - gRPC
  - REST
- Creating a web services using .NET core
- Introduction to multithreading



# Course overview: 11,12 lecture

- Async/await
- Improving application:
  - Debug
  - Diagnostics and profiling
  - Events (operating system)
- Agile
- Functional and non-functional requirements
- Interceptors and middleware in .NET Core



# Course overview: 13,14 lecture

- Introduction to User Experience
- GUI-based application development
- Presentation of the developed software system
- Testing:
  - Unit tests, integration tests and component tests
  - Testing tools and strategies



# Course overview: 15,16 lecture

- Introduction to design patterns (MQ, CQRS etc.)
- Analysis of modern OO systems
- Preparation for the exam and taking the final exam (written)





# Course overview: 17 lecture





# Evaluation

- Exam in written form (max 5.0):
  - Test, semi open and open questions
  - **Exam is considered to be passed if at least 1.5 out of 5 points are collected**
  - **Exam can be taken only when total amount of points collected during the semester is 3.0 or more**
- Laboratory assignments (max 5.0)
  - Three (1.5 + 2.0 + 1.5)
- Additional points (max 1.5)
  - Kahoots during the lecture (<1.0).
  - Advance settlement of tasks (5.0 x 10%).



# What's are the problems with traditional teaching?

- Failure to code in a team:
  - Code reviews
  - Managing work tasks
- Weak personal preparation:
  - Lack of empathy
  - Lack of creativity
  - Minimalism (I do only what I am told to)
- Teaching methods:
  - One to everyone



*„You think education is expensive...  
Try estimating the cost of ignorance“*

Howard Gardner

# What do we do?

- Three apps
- Working in teams
- Every week is like a real development sprint where at the end of it you will give points to each other (more on that during practise)

Teamwork



Working solo





# The apps

- Topic - better tools for education:
  - a. App that helps you to prepare for an exam.
  - b. App that lets you collaborate with your course mates on taking lecture notes.
  - c. Interactive app for the lecture (quizzes and such).
  - d. Your idea?



# Team selection

- Rule of thumb – practise lecturer decides how teams are assembled;
- Recommended approaches:
  - Choose the app you want to build and group up with others who chose the same app;
  - Join students who you like working with;



# 1st practice assignment (1)

- **Official goal:** to develop an app while working in groups while using material covered in 1-6 lectures.
- **Unofficial goal:** to develop an app in ASP.NET / WPF / MAUI, that has basic functionality of assigned APP. Understand the usage of GitHub, coding in team principles. Prepare foundation for future work. Learn to code review yourself and take the feedback when getting one.





# 1st practice assignment (2)

## Requirements:

1. Creating and using your own class, struct, record and enum;
2. Property usage in struct and class;
3. Named and optional argument usage;
4. Extension method usage;
5. Iterating through collection the right way;
6. Reading from a file using a stream;
7. Create and use at least 1 generic type;
8. Boxing and unboxing;
9. LINQ to Objects usage (methods or queries);
10. Implement at least one of the standard .NET interfaces (IEnumerable, IComparable, IComparer, IEquatable, IEnumerator, etc.)



# 2nd practice assignment (1)

- **Official goal:** continue developing an app while working in groups while using material covered in 7-10 lectures.
- **Unofficial goal:** start using database and dependency injection. Deepen the functionality of application while being creative.



# 2nd practice assignment (2)

## Requirements:

1. Relational database is used for storing data;
2. Create generic method, event or delegate; define at least 2 generic constraints;
3. Delegates usage;
4. Create at least 1 exception type and throw it; meaningfully deal with it; (most of the exceptions are logged to a file or a server);
5. Lambda expressions usage;
6. Usage of threading via Thread class;
7. Usage of async/await;



## 2nd practice assignment (3)

8. Use at least 1 concurrent collection or Monitor;
9. Regex usage;
10. No instances are created using 'new' keyword, dependency injection is used everywhere;
11. Unit and integration tests coverage at least 20%;



# 3rd practice assignment (1)

- **Official goal:** implementing web service, understanding of ORM, continue developing an app while working in groups while using material covered in 11-14 lectures.
- **Unofficial goal:** Work on UI, while thinking about end-user experience. Deepen the functionality of application while being creative. Have a ready MVP version of your application.



# *Minimum Viable Product*

A minimum viable product, or MVP, is a product with enough features to attract early-adopter customers and validate a product idea early in the product development cycle.



# 3rd practice assignment (2)

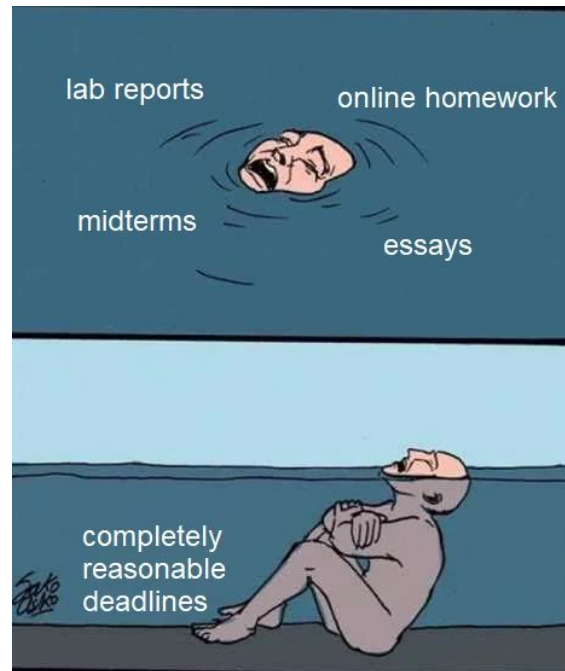
## Requirements:

1. Web service implemented and used;
2. Entity Framework and code-first migrations usage and understanding of difference between
3. code/model/database first approaches;
4. Usage of middleware and at least one interceptor;
5. Unit and integration tests coverage at least 50%;
6. Hackathon like pitch for the application is made;



# Deadlines, but as a calendar

Number	Date
1	2023-09-04
2	2023-09-11
3	2023-09-18
4	2023-09-25
5	2023-10-02
6	2023-10-09
7	2023-10-16
8	2023-10-23
9	2023-10-30
10	2023-11-06
11	2023-11-13
12	2023-11-20
13	2023-11-27
14	2023-12-04
15	2023-12-11
16	2023-12-18







# Practice assignments (1)

- Practice examination is being held during practice lessons. Virtual examination (e.g via email) is not possible.
- Every assignment that is done in time and without any flaws is marked with maximum possible points
  - Flaws make grade lower
  - It is always better to make adjustments as long as there are no flaws
- Lateness leads to the decrease of the maximum assessment by 20% of every delayed week



# Practice assignments (2)

- Additional points are added to the assessment if the work is presented before the deadline (no more than 10% of the final assessment and 5% for every preliminary week).
- During the practice assignment examination lecturer makes sure that students **understand and have ability to change the program.**
- When grading, lecturer checks if version control was used and checks **pull request code reviews.**
- Students are informed about their mark in the same lecture after the examination.



# Practice assignments (3)

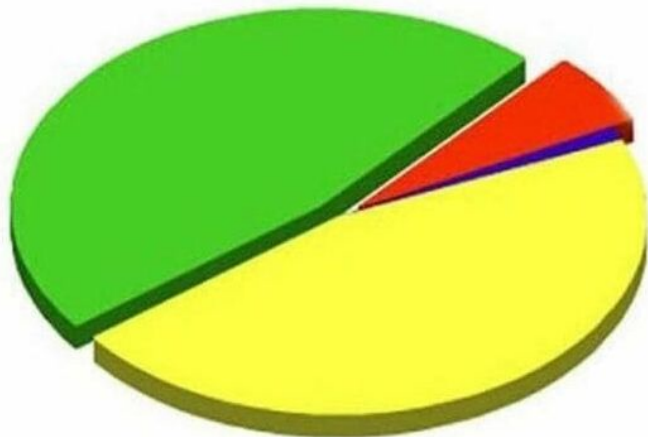
- The point is "pretending" to write code as if you were working for a company, rather than reporting to a university. So :
  - Your code will be reviewed (pull requests),
  - Code will be long living after the lectures,
  - You don't want to be that person, who's name is being mentioned in the context of: „who coded this peace of nonsense???",
  - Basically – you have to be happy with the code that you write.





# Practice assignments (4)



# Practice assignments (5)

## WHAT I LEARN FROM GROUP PROJECTS



-  The information
-  How to work with people
-  How to do entire projects on my own
-  How much I hate people

When only you did the work





# Practice assignments (6)

- Will try to keep results updated in Moodle (VMA), but can't promise.



# Literature (1)

- Main: Andrew Troelsen. Pro C# 2010 and the .NET 4
- Additional:
  - J.Skeet. **C# in Depth**.
  - D.Clark. Beginning C# Object-Oriented Programming.
  - J.Purdum. Beginning Object-Oriented Programming with C#.
  - A.Hunt, D.Thomas. **The pragmatic programmer**: from journeyman to master.
  - Scott Allen - C# Fundamentals with Visual Studio 2015  
<https://app.pluralsight.com/library/courses/c-sharp-fundamentals-with-visual-studio-2015/table-of-contents>

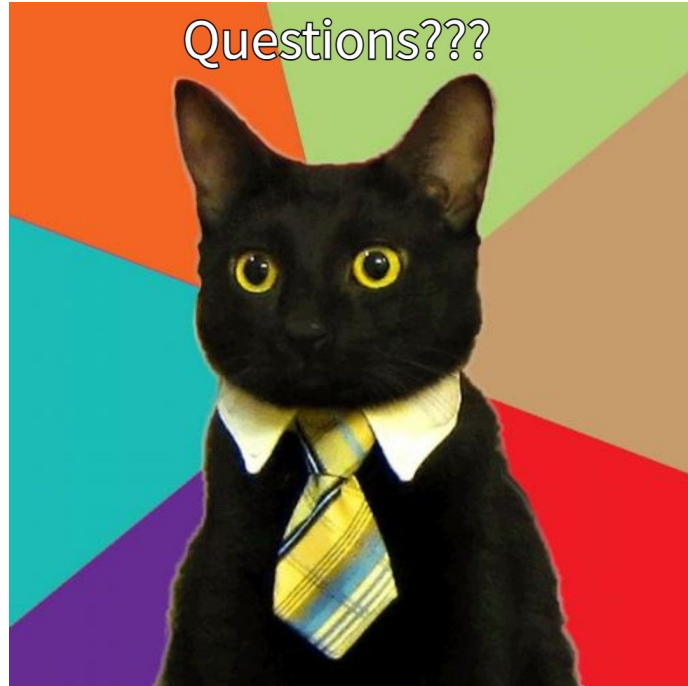


# Literature (2)

- Preparation for exam 70-483:
  - MCSD Certification Toolkit. Exam 70-483. Programming in C#.
  - Programming in C#. Exam Ref 70-483.



# Questions?





# Development IDE

- VS Code (personal recommendation):
  - [Visual Studio Code](#)
- Microsoft Visual Studio
  - Community: recommended and FREE
  - Ultimate/Pro: Microsoft Imagine.
- JetBrains Rider
  - [Rider: The Cross-Platform .NET IDE from JetBrains](#)
  - [programine iranga \[IT wiki\]](#)



# What is Git?

Short workshop on Git and GitHub



**GITHUB**



**DROPBOX**



v1.0.0



v1.0.1



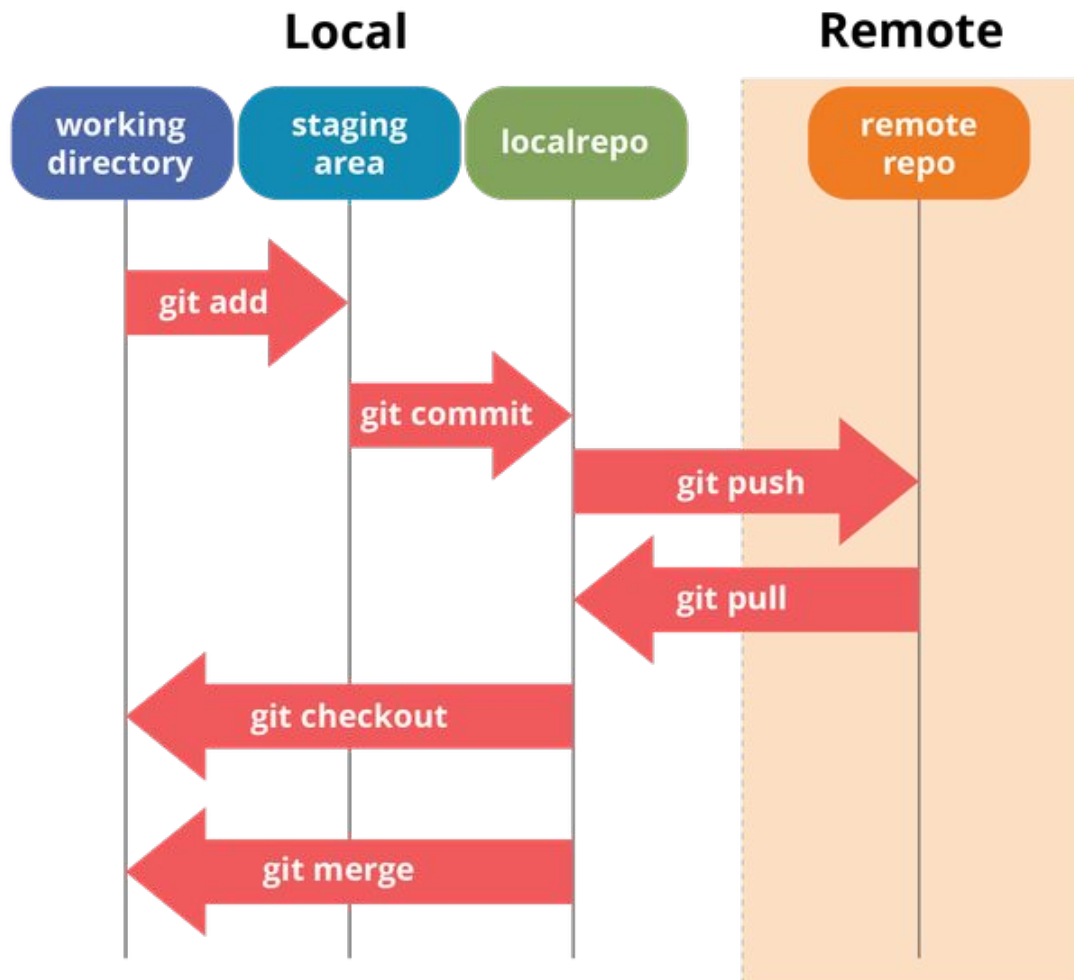
# Agenda

- <https://git-scm.com/>
- Learn what is Git (and version control system) and what it is used for
- Learn git basic commands and try them out while working on a project
- Learn to create a GitHub repository and use it for your own projects
- Learn to use Git and GitHub to enhance workflow when working in teams
- Look into how Git and GitHub play out there in the wild (a peek into enterprise infrastructure)



# What is Version Control System (VCS) for?

- Tracking project changes
- Collaboration in teams and between teams in a project
- Getting out of every screw up you made
- Git is a VCS





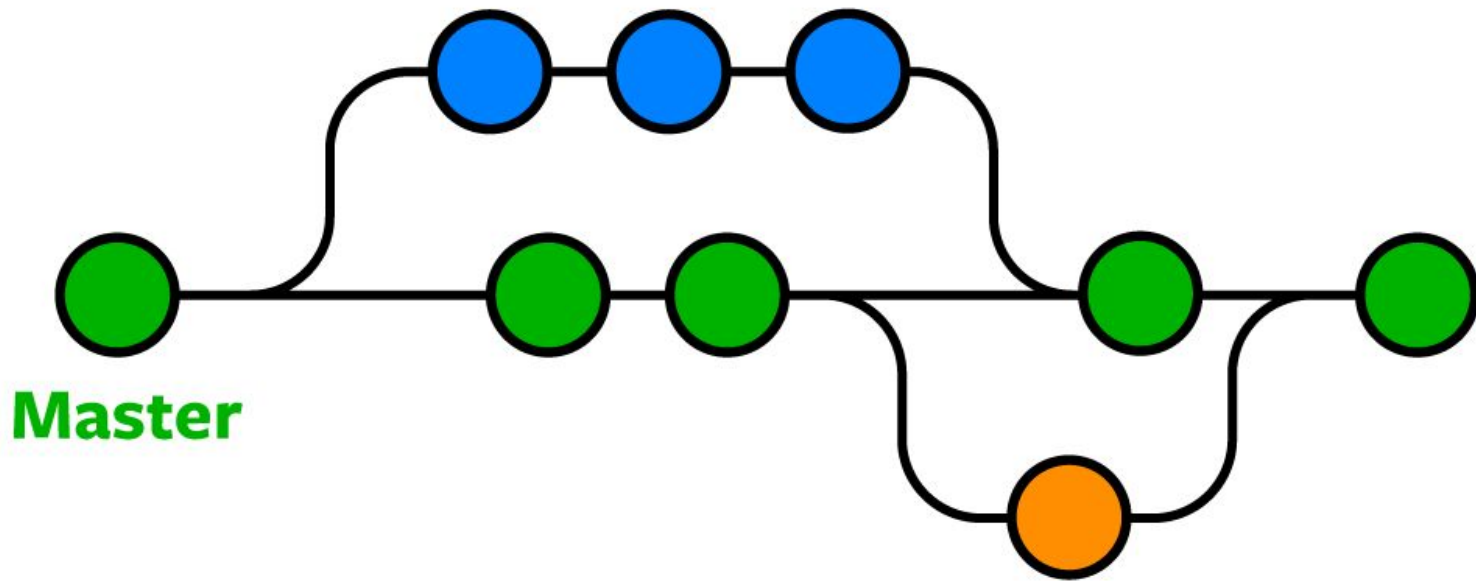
Let's talk about teamwork





# Branching

**Your Work**



**Someone Else's Work**



# Code reviews

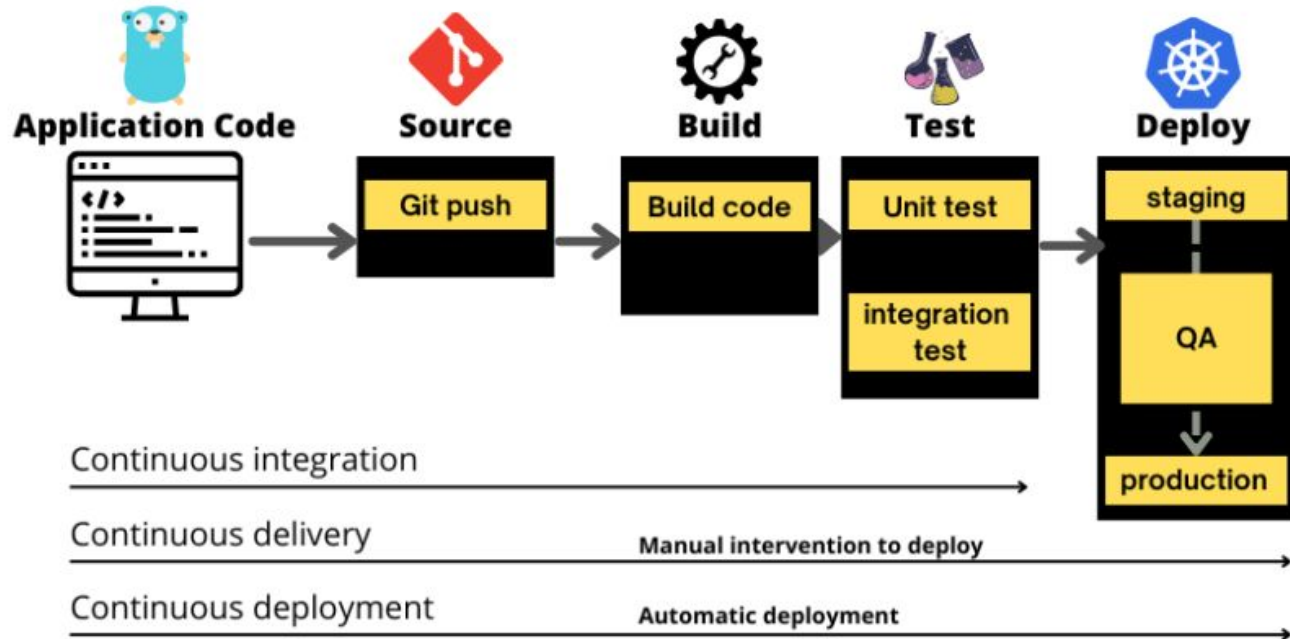
- Bug prevention
- Learning / Breaking bad habits
- Improve code quality



# Good practices when reviewing code

- Keep PR as small as possible
- Foster a positive code review culture
- Leave constructive feedback
- Take your ego out of the equation

# CI/CD pipelines



# Now you know it

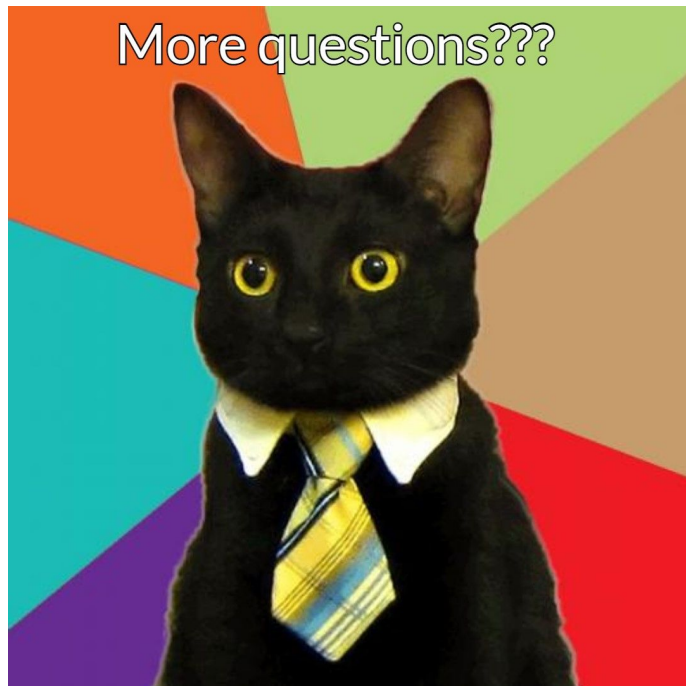




# Final advice

- Keep your commits small and commit often
- Write descriptive commit messages
- Keep the welcoming attitude during code reviews
- Think about people reviewing your code
- Remember that Git is NOT the same as GitHub
- Git essentials in short: <https://rogerdudler.github.io/git-guide/>
- <https://learngitbranching.js.org/>
- For those who like hardcore stuff: <https://git-scm.com/docs>
- Git stages and more commands:  
<https://ndpsoftware.com/git-cheatsheet.html>

# Questions?





# Next time

- We will go through the basics of C# and .NET