

# Visual Studio og tools

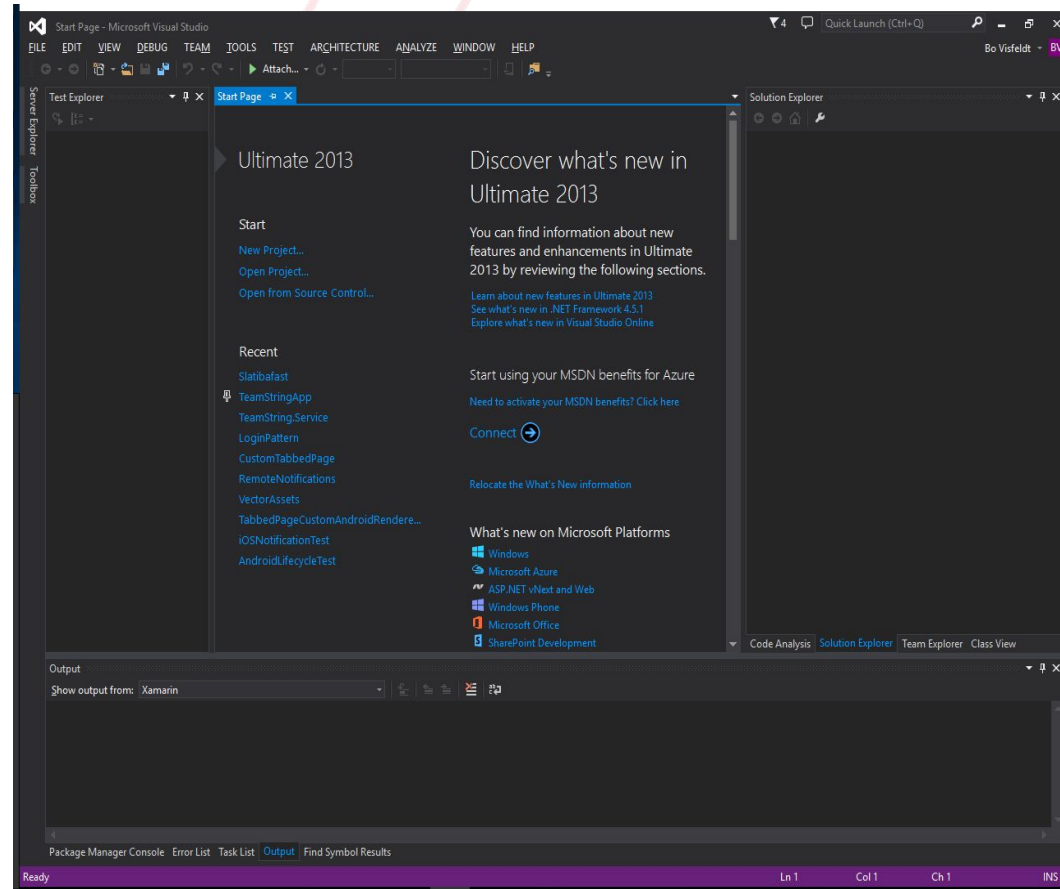
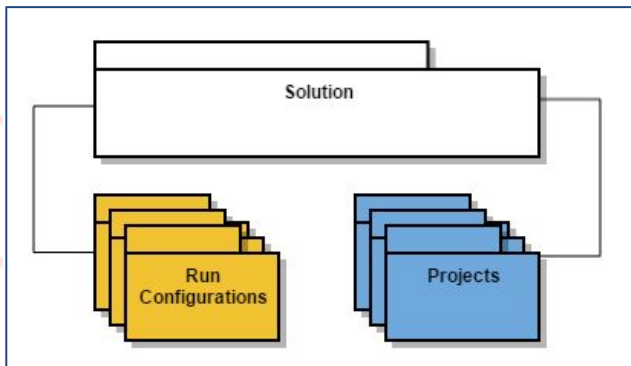
## Xamarin Mobil Udvikling Modul 2

# Visual Studio, kort gennemgang 1/3.

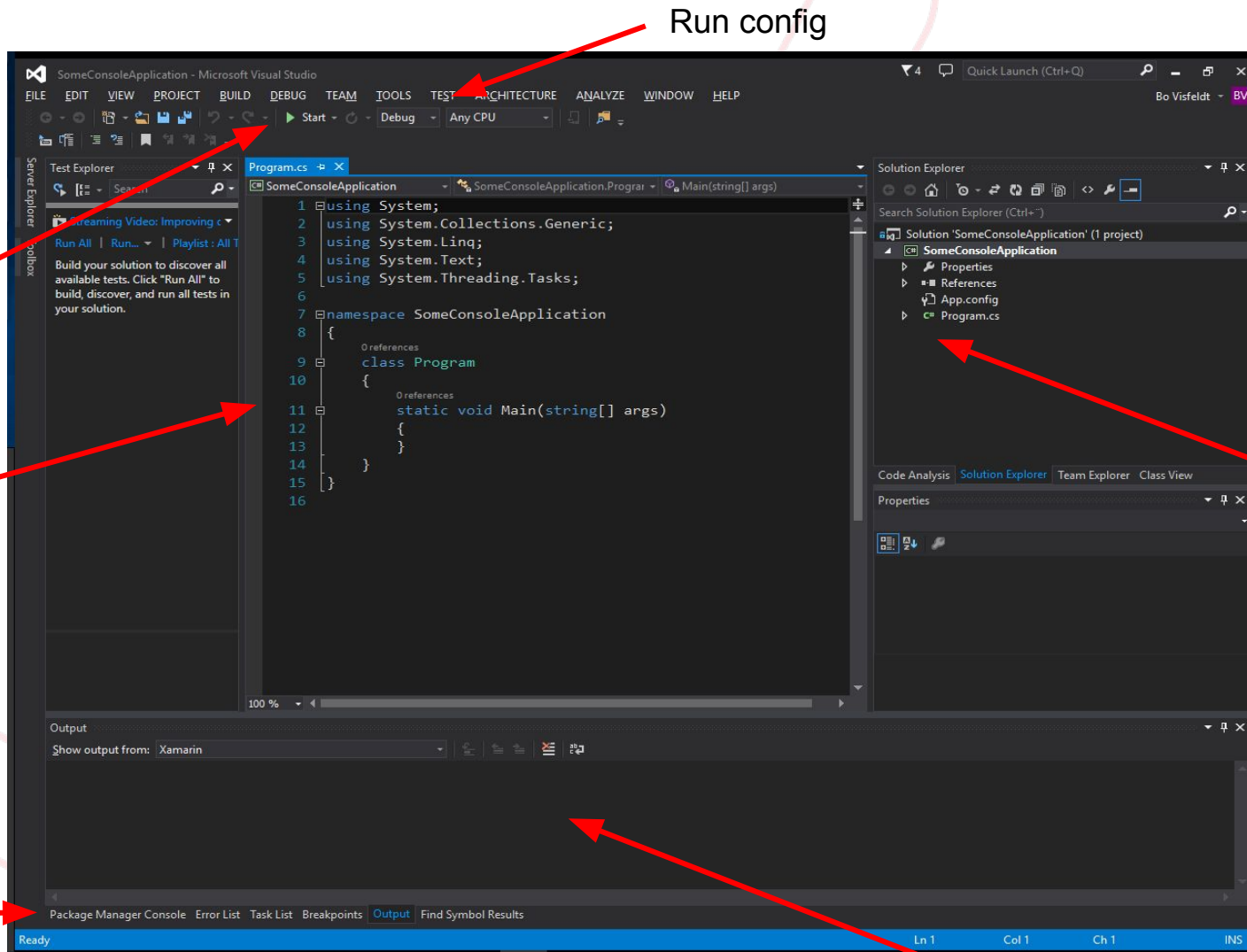
I 2015 er Xamarin nu inkluderet.  
(via [Xamarin.com/download](http://Xamarin.com/download))

Forskel på Solutions og projekter.

Run Configurations.



# Visual Studio, kort gennemgang 2/3.



# Visual Studio, kort gennemgang 3/3.

Nyttige shortcuts:

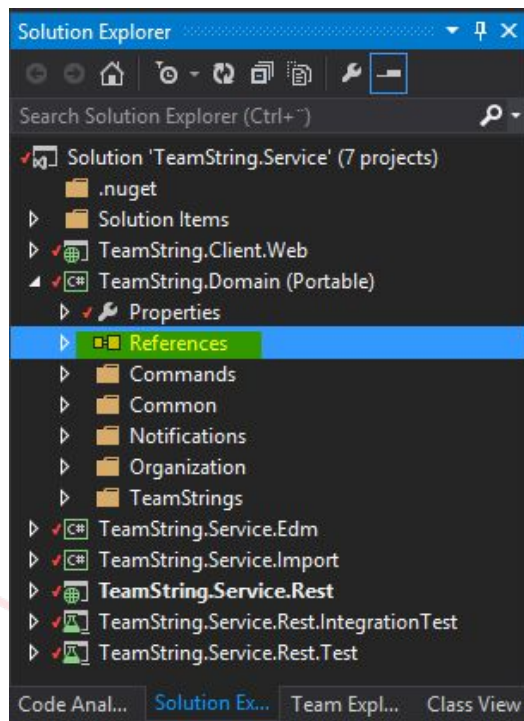
- Go to definition: F12
- Navigate to: Ctrl+,
- Show smart tag (Auto import): Ctrl+.
- Run all tests in context: Ctrl+r, t
- Debug all tests in context: Ctrl+r, Ctrl+t
- Clean all code: Ctrl+k, Ctrl+d
- Save All: Ctrl+Shift+s
- F5: Run Startup Project
- F6: Compile Solution

# NuGet

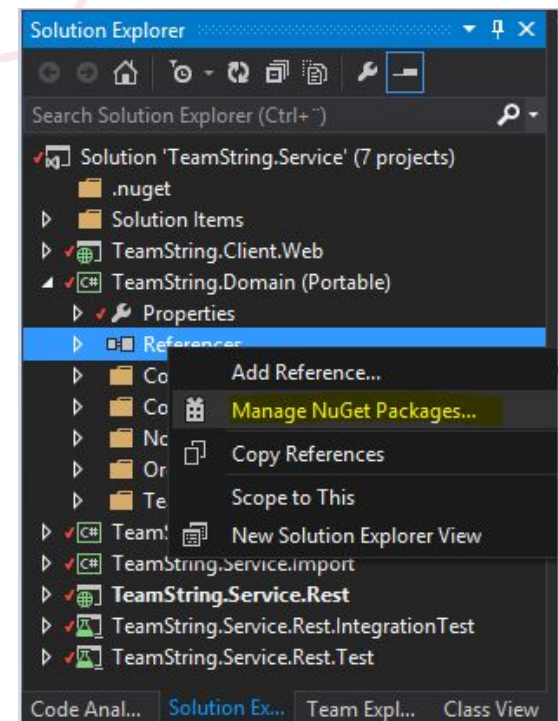
- Package manager for libraries.
- Bruges til nemt at installere afhængigheder for et projekt.
- Eksempler: NUnit, Azure client libs, JSON parser, HTTP klient, ...
- Kæmpe repository af pakker på <http://www.nuget.org>.
- Tæt integreret med Visual Studio.

# NuGet - i VS2015

Tilføje pakke til et projekt:

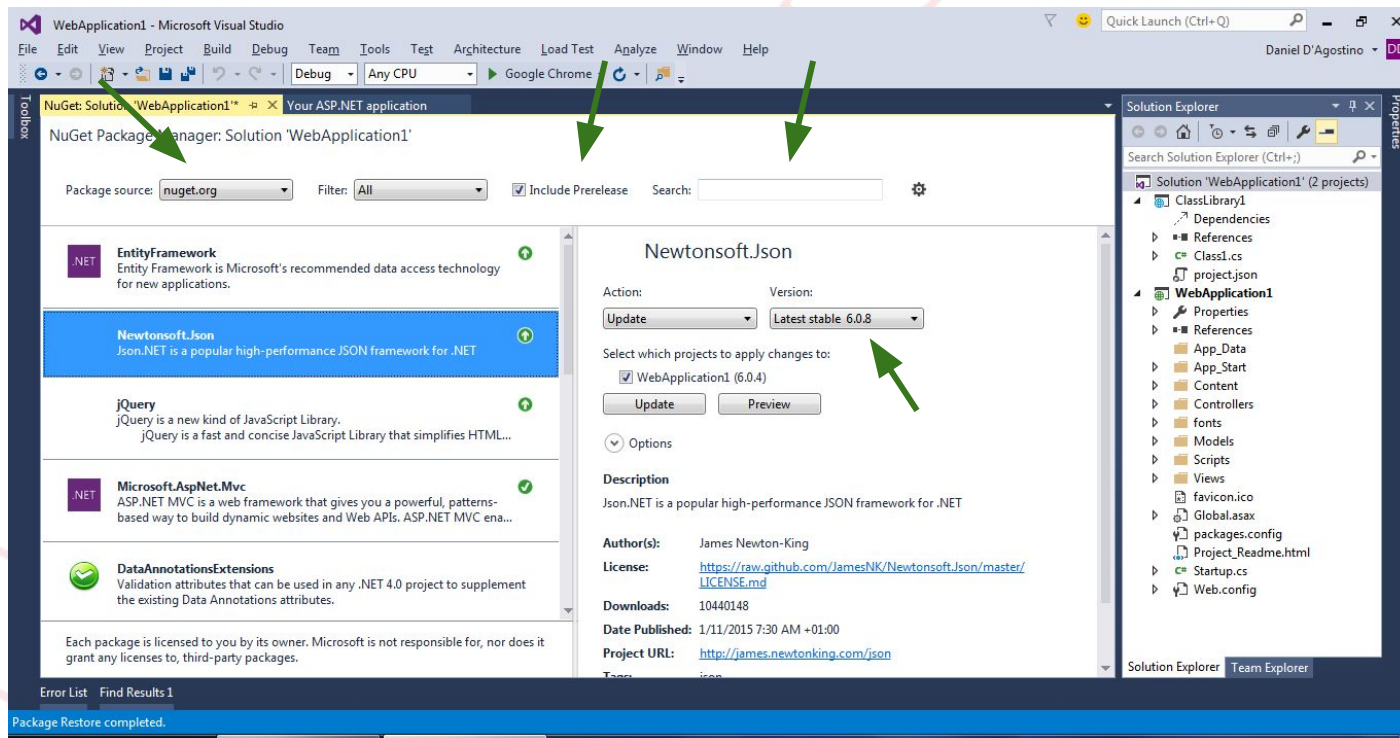


Højreklik



# NuGet - i VS2015

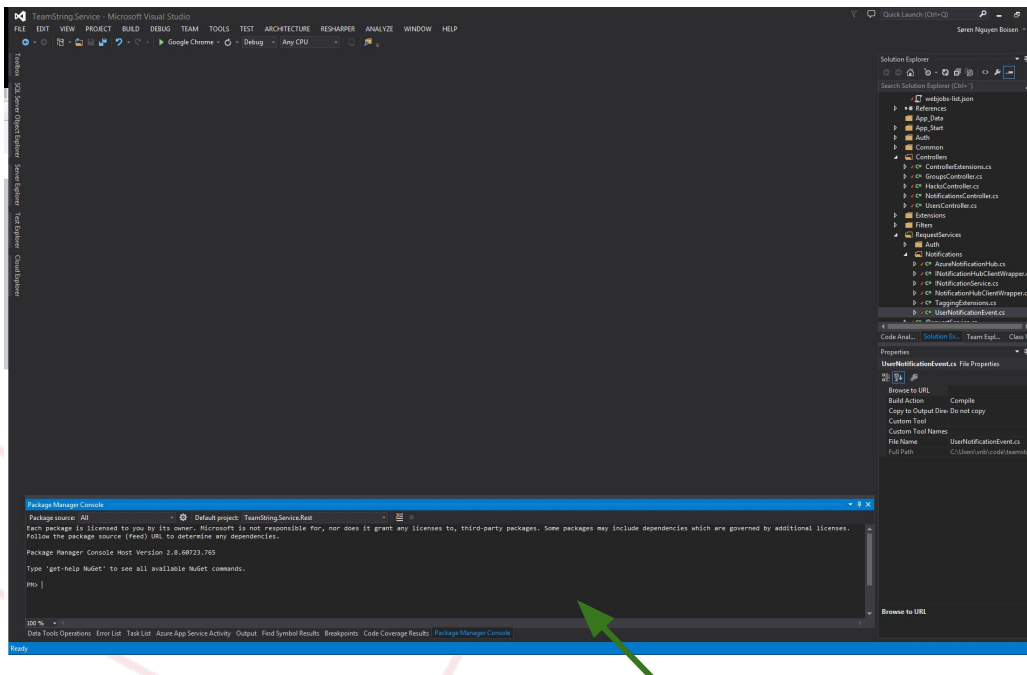
## NuGet Visual Studio UI:



# NuGet - i VS2015

## Package Manager Console (kommandolinie):

- Åbne:
  - Tools -> NuGet Package Manager -> Package Manager Console

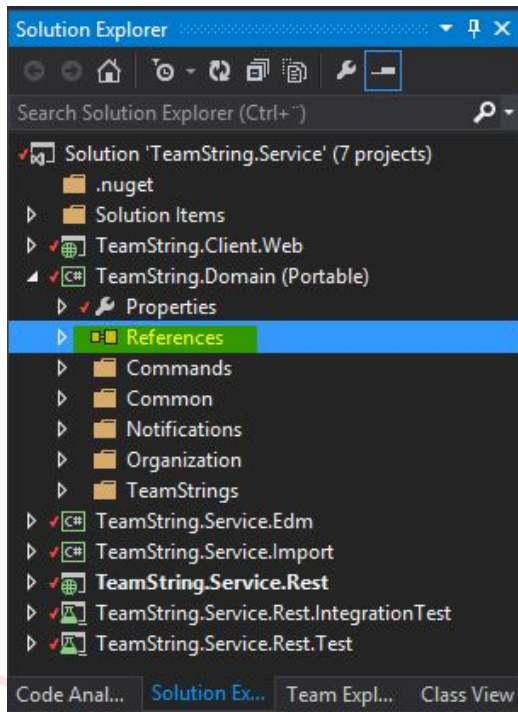




## NuGet - bemærkninger I

- NuGet installerer automatisk afhængigheder, når man installerer en pakke i et projekt.
- MEN: NuGet installerer IKKE pakken i de projekter, som afhænger af projektet, man installerer pakken i.

## NuGet - bemærkninger II



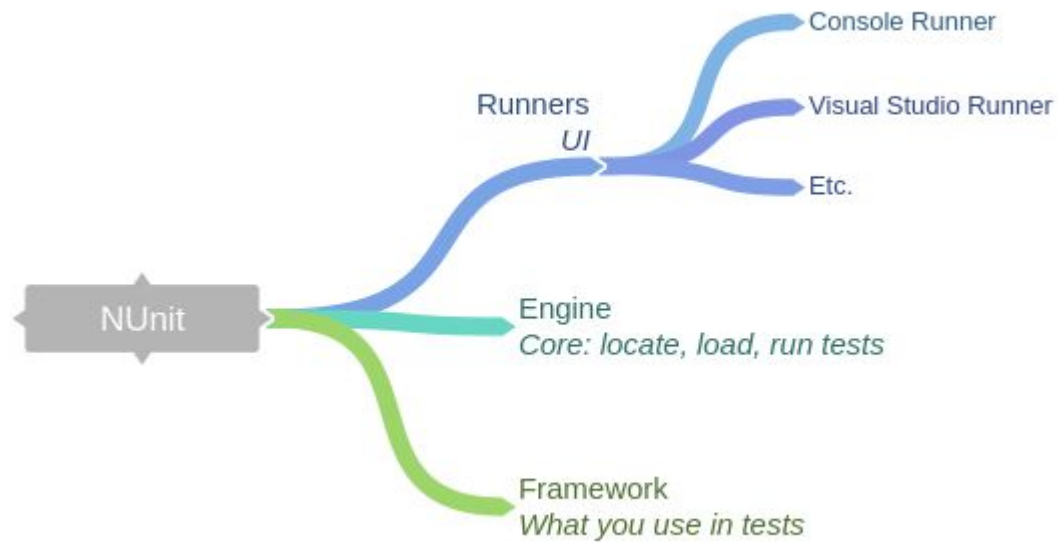
- Alle andre projekter i solution afhænger af TeamString.Domain.
- Hvis jeg installerer en NuGet pakke i TeamString.Domain, er jeg nødt til manuelt at installere den i alle de andre projekter.

# NuGet - Opgave

# NUnit

- Test framework - ikke kun til unit tests.
- NUnit 3.0 kan køre tests parallelt og har god understøttelse for data-drevne tests.
- Med nunit.xamarin kan tests køres on-device.
- [TestFixture] attribut på en klasse indikerer at den indeholder tests.
- [Test] på en metode indikerer at det er en test case.

# NUnit 3.0 - Arkitektur



# NUnit - Struktur



# JUnit - Skrive tests

```
using System;
using NUnit.Framework;

namespace Writing.NUnit.Tests
{
    [TestFixture]
    public class MyTest
    {
        [Test]
        public static void TestCase01()
        {
            throw new NotImplementedException();
        }
    }
}
```

## NUnit - Genveje

- Ctrl+R,T for at køre:
  - Aktuel test case, hvis markør indenfor metoden.
  - Alle test cases i filen, hvis markør udenfor cases.
- Ctrl+R,Ctrl+T for at debugge.
- TestExplorer.ShowTestExplorer (ikke bundet).



# NUnit - Opgave

## Basal Git

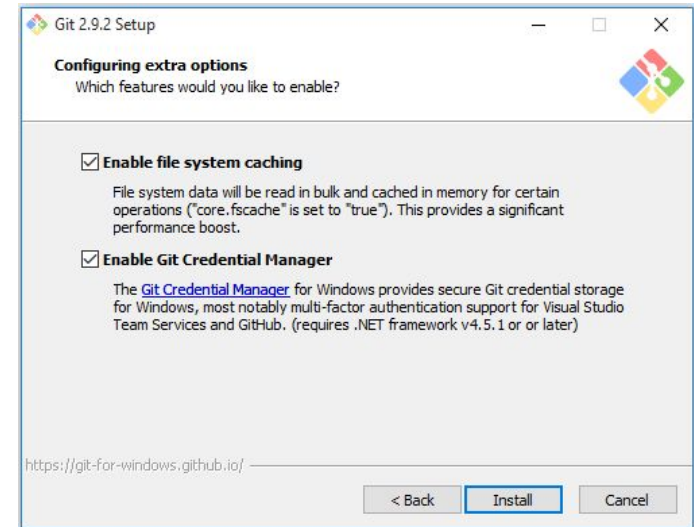
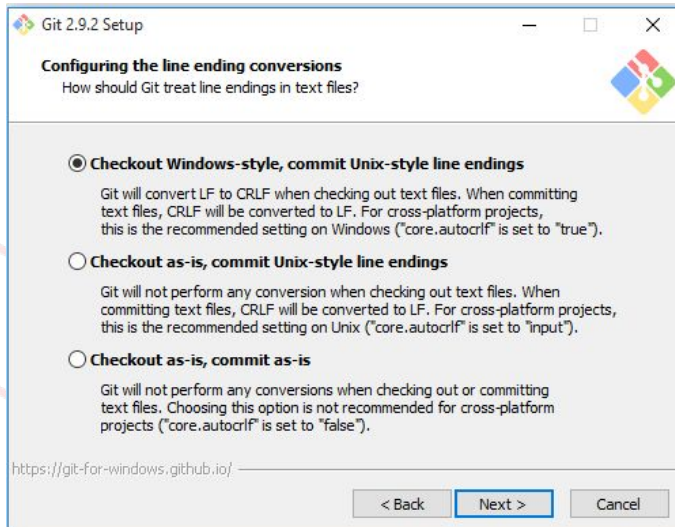
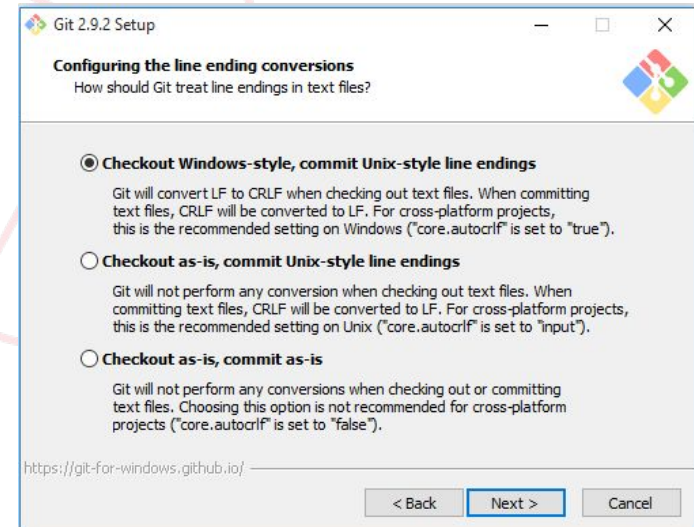
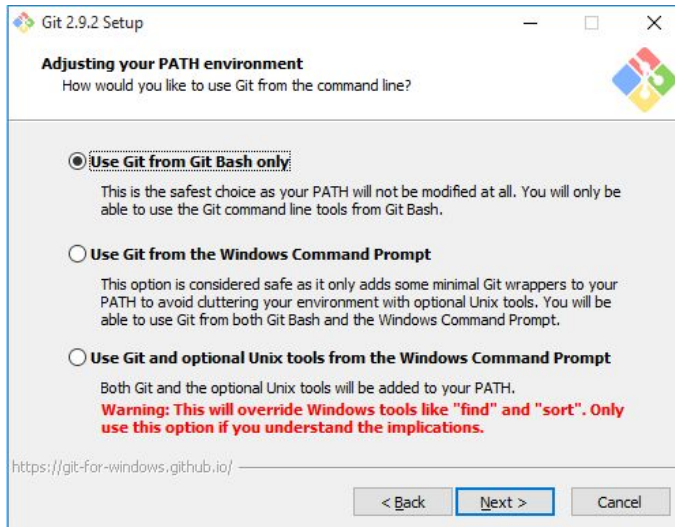


Download og installer git: <http://msysgit.github.io/>

- Open Source Version Control System.
- Oprindeligt til \*nix systemer, men porteret til Windows (med enkelte irritationer)
- Der er mange repository udbydere. GitHub og Bitbucket er at nævne.
- Bør læres fra konsollen, men der er plugins til de fleste IDE'er

Simpel intro: <http://rogerdudler.github.io/git-guide/>

# Installer Git



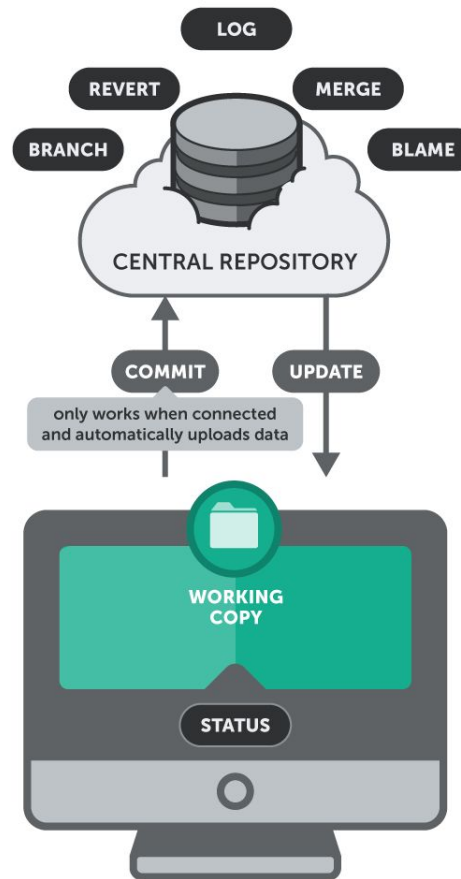
Download og installer git: <http://msysgit.github.io/>

Simpel intro: <http://rogerdudler.github.io/git-guide/>

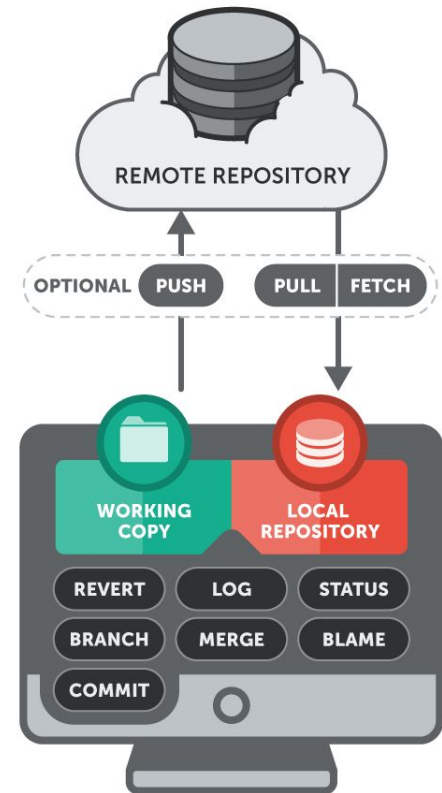
# Hvad er forskellen på andre VCS og Git

- CVS og SVN er centraliseret
- Git er Decentraliseret.
- I SVN kan man ikke comitte når man er offline da transfers foregår autmatisk.
- I SVN er alle filændringer med. I Git siger du hvilke filer du vil have med.
- SVN benytter versionsnumre. I Git er dette ændret til en hash værdi da udviklere committer offline.
- Alle klienter er en potentiel Git server.
- I Git foregår konflikter som regel lokalt og ikke på serveren.

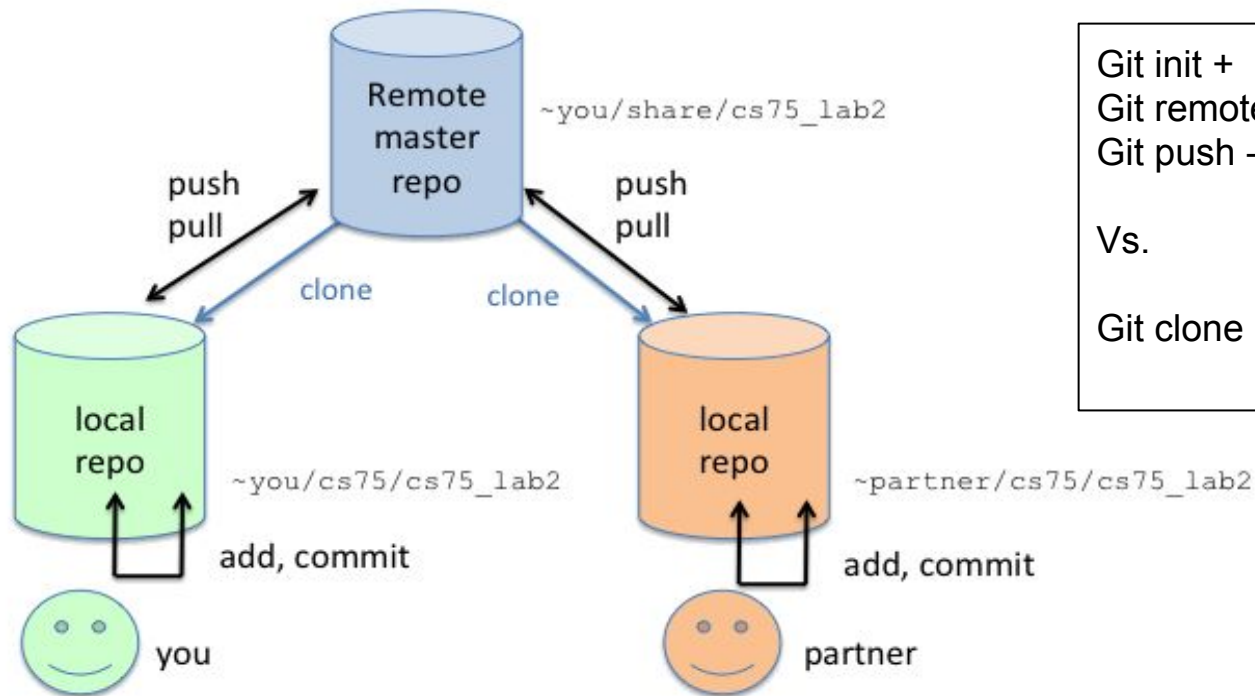
## SUBVERSION



## GIT



# Hvordan fungerer Git?



Git init +  
Git remote add +  
Git push -u origin master

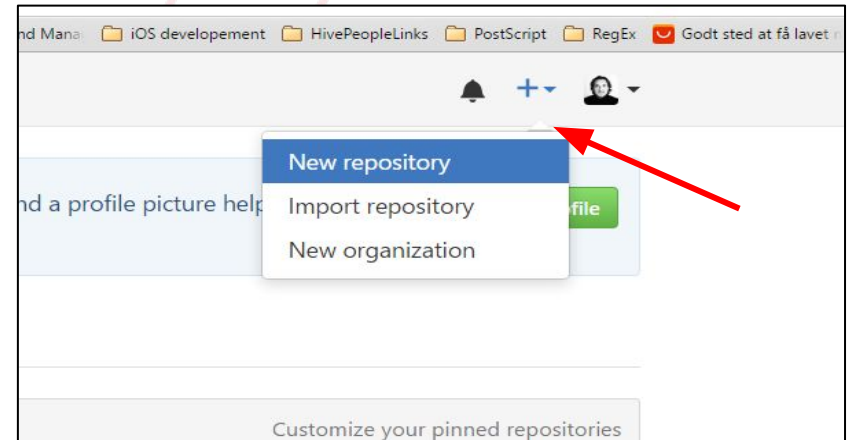
Vs.

Git clone

- Github er et sted hvor man gratis kan have repositorier liggende til open source. Du kan betale for private.
- BitBucket er gratis for virksomheder op til 5 brugere, men ikke lige så nice som GitHub.

# Lav et GitHub Repos 1/2

1. Opret en konto på GitHub.
2. Opret et repository




## 2.1 Giv det et funky navn

- Marker som public hvis du ikke vil betale en blødende formue for private

## 2.2 kopier den ssh sti du får til repos'et

### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner  
 bovisfeldt

Repository name

Great repository names are short and memorable. Need inspiration? How about [ideal-octo-barnacle](#).

Description (optional)

☒ Public  
Anyone can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

<http://rogerdudler.github.io/git-guide/>



# Lav et GitHub Repos 2/2

3. Lav et rod directory for alle dine repos. (vi foreslår c:\code)

4. Åben git-bash

5. Gå til dit nylavede directory. Husk at git-bash har unix syntax, og path'en således er "**cd /c/code**"

6. Klon det repos du lavede før og gå ind i directoriet

```
bvi_hivepeople@DESKTOP-IC612EE /c
$ cd /c/code
bvi_hivepeople@DESKTOP-IC612EE /c/code
$ git clone https://github.com/bovisfeldt/i-hate-bongo-drums.git
Cloning into 'i-hate-bongo-drums'...
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
bvi_hivepeople@DESKTOP-IC612EE /c/code
$ ls
akutjournalprinter  asyncmodel  i-hate-bongo-drums  teamstringapp  tea
bvi_hivepeople@DESKTOP-IC612EE /c/code
$ cd i-hate-bongo-drums/
bvi_hivepeople@DESKTOP-IC612EE /c/code/i-hate-bongo-drums (master)
$
```

# Filer i dit repos

**git status**

(viser hvilke filer der er staged og generel status for DIT LOKALE repos)

**git add <filename>**

(adder en fil til staging area [the index])

**git add \***

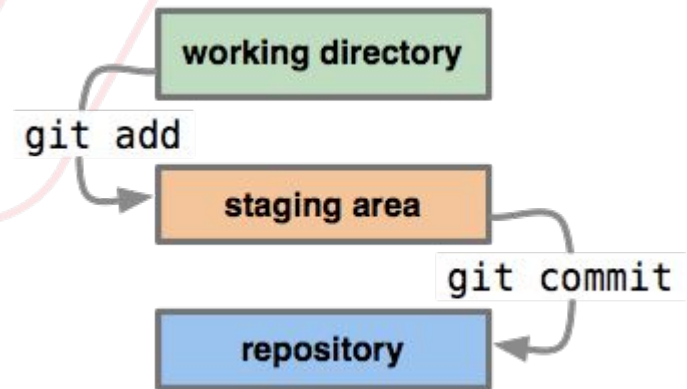
(adder alle ændrede filer til staging area)

**git commit**

(comitter alle filer der er i staging area til repos.  
Åbner først VI så du kan skrive en commit besked)

**git commit -m "message"**

(comitter alle filer der er i staging area til repos med beskeden givet)





# Filer i dit repos

git pull

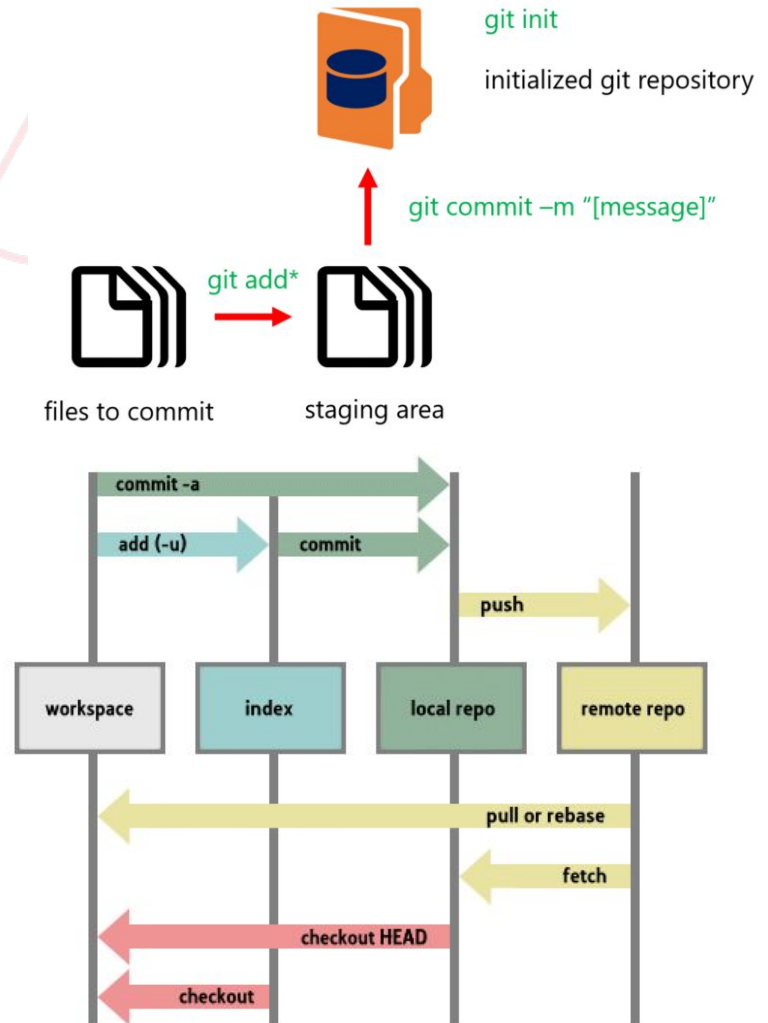
(hiver ændringer fra upstream repos ned til dit repos)

git push

(pusher dit repos til remote upstream repos og merger filerne)

git fetch

(hiver nye branches ned... pt arbejder I kun med en branch hver)



## Opgave:

1. Installer GIT
2. Opret konto på github
3. Opret et repository på github
4. Klon det tomme repository
5. Put tre nye tekst filer i
6. Tilføj disse til dit lokale repos.
7. Push dem til Github
8. Verificer på GitHub at de nye filer er der.
9. Lav ændringer til en af filerne
10. Check “git status”
11. Add filen til lokal repos
12. Send til github og verificer ændringen.