**Hands-on lab**

**Lab: Git**

**December 2018**

**Doel**

**Het is de bedoeling om vertrouwd te raken met de basiscommando's van Git via de command prompt**

**Opgaven**

**Lees de slides nog eens door en lees de tekst “Git Succintly”. Eventueel kan je ook de demo's op Pluralsight (Git Fundamentals) bekijken.**

**Lokaal werken met git**

1. **Start met een Java of .NET project in een bepaalde folder. Compileer en run het project zodat je het programma minstens éénmaal hebt uitgevoerd.**

*/*

1. **Maak nu van deze folder een working directory door via de prompt het gepaste git commando in te geven.**

*git init*

1. **Voeg nu een gepaste configuratiestap door zodat overbodige bestanden en mappen genegeerd zullen worden. Tip: gitignore.io**

*code .gitignore*

*>>>* # Created by https://www.gitignore.io/api/dotnetcore

# Edit at https://www.gitignore.io/?templates=dotnetcore

### DotnetCore ###

# .NET Core build folders

/bin

/obj

# Common node modules locations

/node\_modules

/wwwroot/node\_modules

# End of https://www.gitignore.io/api/dotnetcore

*git add .gitignore*

1. **Voeg nu de overblijvende bestanden toe en doe een eerste commit.**

*git add .*

*git commit -m “Added project files”*

1. **Start een nieuwe ontwikkelbranch voor het toevoegen van een nieuwe feature aan het programma. Commit een aantal keer. Experimenteer hierbij met de staging area: niet alle gewijzigde files hoeven bij de eerstvolgende commit meegenomen te worden.**

*git branch develop*

*git checkout develop*

*code User.cs*

*code Book.cs*

*code Building.cs*

*git add User.cs*

*git commit -m “Added User class”*

*git add .*

*git commit -m “Added other classes”*

1. **Merge dit nu met de master branch. Welk type van merge is dit?**

*git checkout master*

*git merge develop*

*>>> fast-forward*

1. **Verwijder nu deze ontwikkelbranch.**

*git branch -D develop*

1. **Start opnieuw een ontwikkelbranch en commit een aantal keer.**

*git branch develop2*

*git checkout develop2*

*code Vase.cs*

*code Chimney.cs*

*git add Vase.cs*

*git commit -m “Added Vase class”*

*git add .*

*git commit -m “Added other classes”*

1. **Switch nu opnieuw naar de master branch en commit ook daar een aantal keer, waarbij je zeker minstens éénmaal in eenzelfde bestand als in de ontwikkelbranch commit.**

*git checkout master*

*code Vase.cs*

*git add Vase.cs*

*git commit -m “Changed Vase”*

1. **Merge nu alles naar de master branch. Welk soort merge is dit?**

*git merge develop2*

*code Vase.cs*

*git merge develop2???*

1. **Verwijder de ontwikkelbranch.**

*git branch -D develop2*

1. **Opnieuw dezelfde oefening: development branch maken, committen en ook in master committen. Doe nu een rebase in plaats van een merge. Wat is nu het resultaat?**

git init

code Test.cs

git add .

git commit -m “test”

git branch develop

git checkout develop

code Dev.cs

git add Dev.cs

git commit -m “Added Dev.cs”

git checkout master

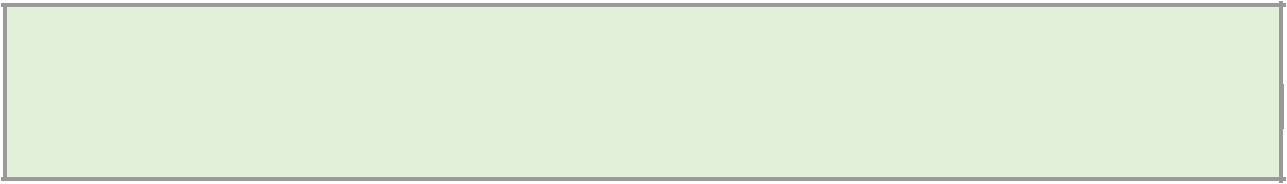
code Mas.cs

git add Mas.cs

git commit -m “Added Mas.cs”

git rebase develop

>>> changes in current branch will be added on top of changes from the merging (or rebasing) branch



**DOELSTELLING**

**Je moet goed kunnen aangeven wat concepten zijn zoals staging, commit, branch, merge strategieën en rebase.**

**Remote Server op Github, Bitbucket of Azure DevOps**

1. **Push je master branch naar Github, Bitbucket of Azure DevOps repository.**
2. **Nodig een collega uit om die master branch lokaal te downloaden.**
3. **Oefen samen het uitwisselen van code en het oplossen van conflicten.**
4. **Is het zinvol om alle branches die jij lokaal maakt ook te pushen?**
5. **Bedenk zelf voor je eigen teamproject een zinvolle workflow. Welke ALM features van Github en/of Bitbucket vind je zinvol? Experimenteer!**
6. **Doe nu hetzelfde met Azure DevOps**
7. **(optioneel) Experimenteer met Sourcetree.**
8. **(optioneel) Experimenteer met Gitkraken.**

**Enkele geavanceerde scenario's**

1. **Stel je werkt in een branch en hebt enkele commits gedaan, maar bent bezig aan enkele bestanden die nog niet gecommit zijn. Je moet opeens een bugfix doen op de master. Hoe ga je tewerk? (Hint: stash en branch)**

*git add .*

*git stash*

*git checkout master*

*code FileToBugfix*

*git add .*

*git commit -m “Performed bugfix”*

*git push*

*git checkout BranchWithWork*

*git stash pop*

1. **Stel je hebt een nieuwe feature afgewerkt in een aparte branch die je via rebase naar**

**de master wil brengen. Je wil de 5 commits die je gemaakt hebt samenvoegen tot 1 commit met als commentaar “feature X is afgewerkt”.**

**[Referentie]**

>>> squash