

Verwendung GIT mit Unity

Benötigte Vorkenntnisse

- Grundlagen Verwendung von Versionskontrollsysteme
- Grundlagen Verwendung GIT über Shell
- Grundlagen Verwendung Unity
- Verwendung von Gitlab

Ziele

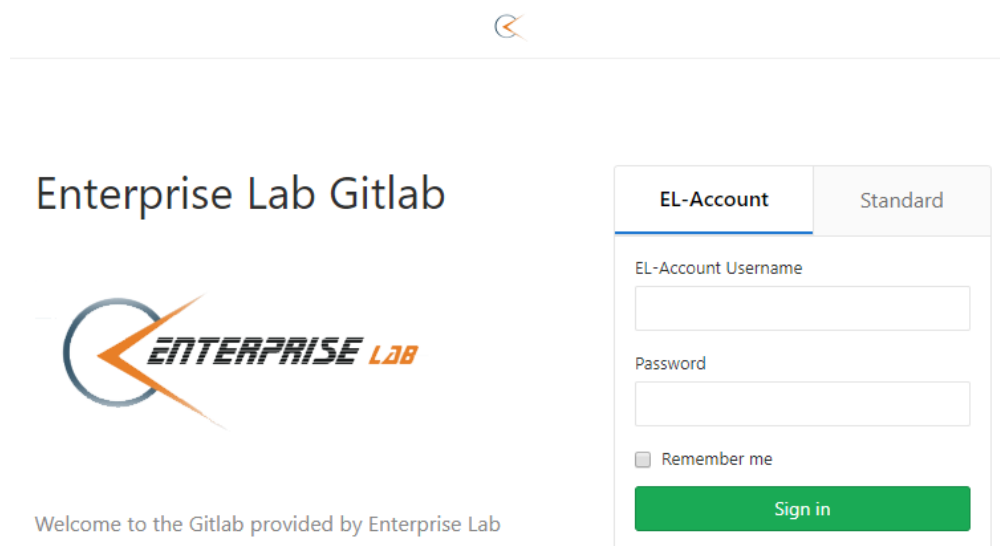
- Verwaltung eines Unity-Projektes in GIT

Anleitung

1. Erstellung eines neuen Repository

Grundsätzlich ist jede Plattform welche GIT anbietet geeignet, in diesem Tutorial verwenden wir Gitlab. Dies da so momentan die Schulprojekte betreut werden.

- a. Bei der Gitlab-Plattform mit deinem Benutzer anmelden



Enterprise Lab Gitlab

Welcome to the Gitlab provided by Enterprise Lab

EL-Account Standard

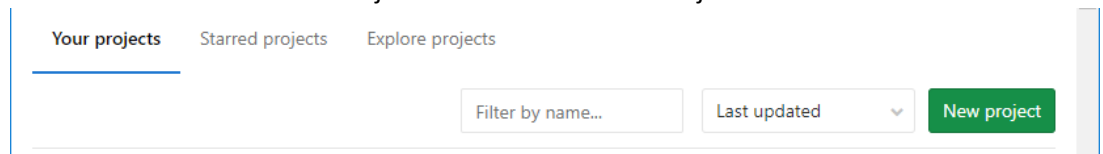
EL-Account Username

Password

☐ Remember me

Sign in

- b. Anschliessend unter «Your Projects» den Punkt «New Project» wählen





Your projects Starred projects Explore projects



Filter by name...

Last updated

New project


- c. Die Standard-Einstellungen so belassen und nur den Projekt-Namen im Feld «Project Name» auf den gewünschten Namen anpassen. Anschliessend auf «Create project» klicken


 Projects





New project


Create or Import your project from popular Git services

Create from template 



Blank



Ruby on Rails



Spring



NodeJS Express


Import project from


 GitHub

 Bitbucket

 Google Code

 Fog

 git Repo by URL

 GitL

OR

Project path

https://gitlab.enterpriselab.ch/tcnussba/


Project name




TutorialRepository

Want to house several dependent projects under the same namespace? [Create a group](#)

Project description (optional)

Description format

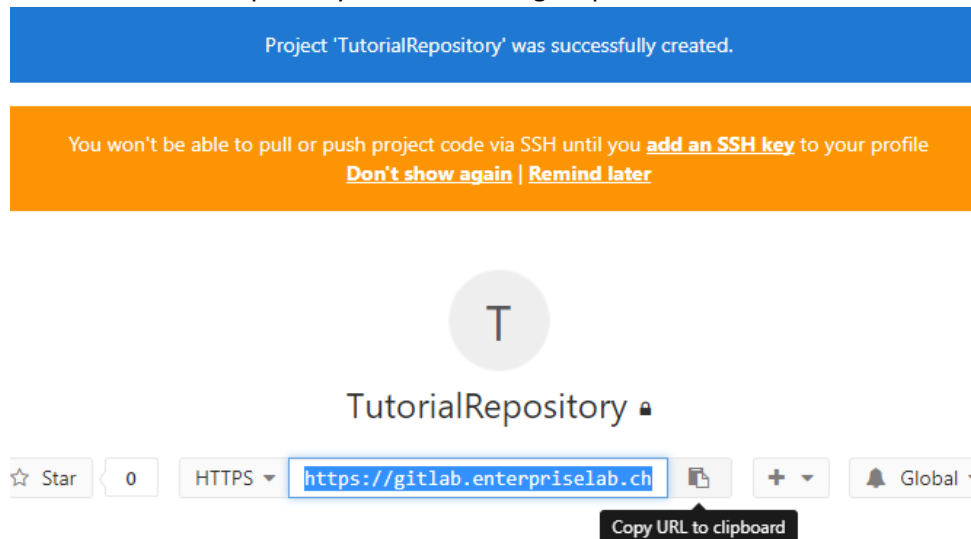
Visibility Level 

- ☒  Private
Project access must be granted explicitly to each user.
- ☐  Internal
The project can be accessed by any logged in user.
- ☐  Public
The project can be accessed without any authentication.

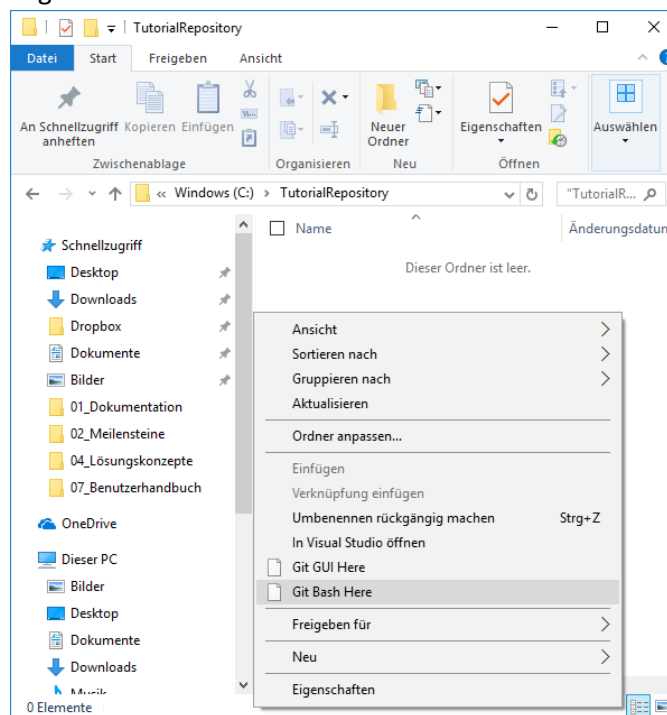
Create project

2. Initialisierung des Projektes

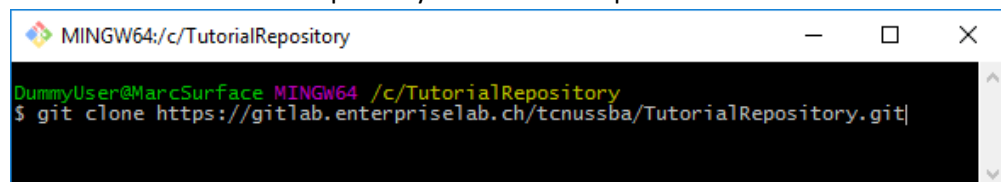
- a. Im neu erstellten Repository die URL wie folgt kopieren



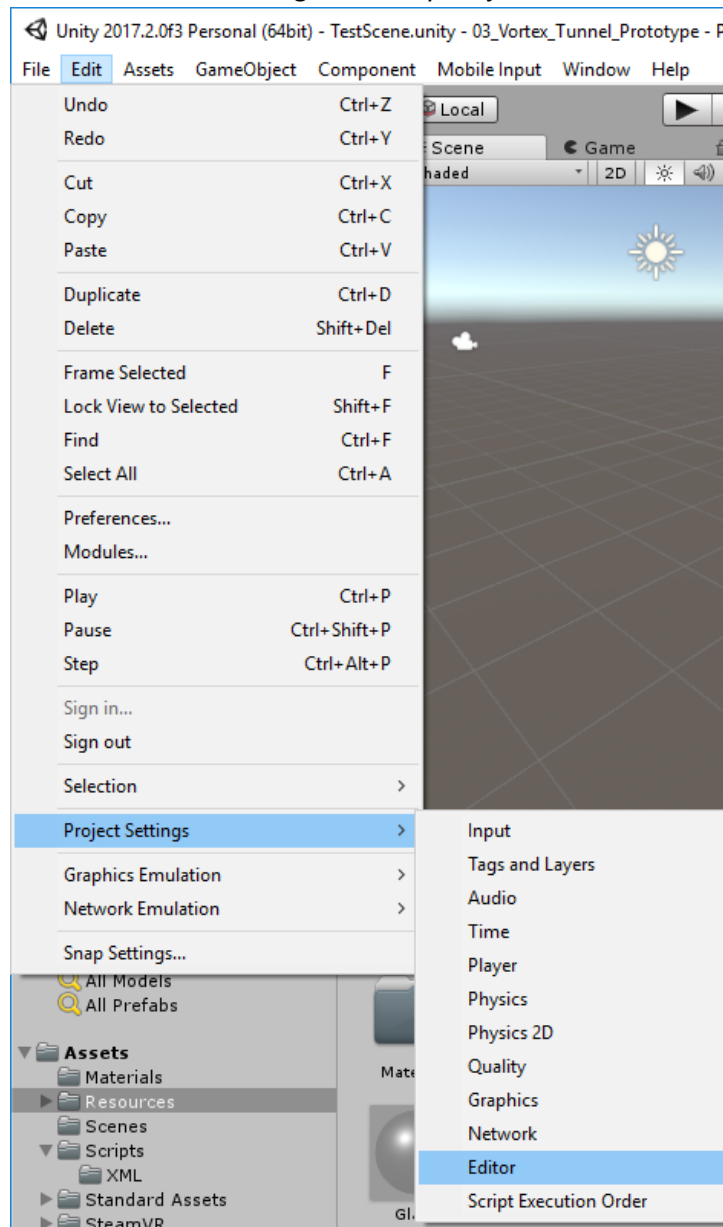
- b. Anschliessen die Git Bash Konsole im Ordner öffnen, in welchem das Projekt später liegen soll



- c. Anschliessend das leere Repository herunterladen per



3. Erstelle oder Kopiere ein Unity-Projekt in den neu erstellten Ordner
 - a. Öffnen die Editor-Settings des Unity-Projektes



- b. Setze unter «Version» den «Mode» auf «Visible Meta Files» und unter «Asset Serialization» den «Mode» auf «Force Text»



Dies ist notwendig, damit Unity diese Änderungen nicht in notwendigen Projektdateien anpasst und somit zu Versionskonflikten führt.

4. Erstellung einer .gitignore-Datei für Unity-Projektdateien

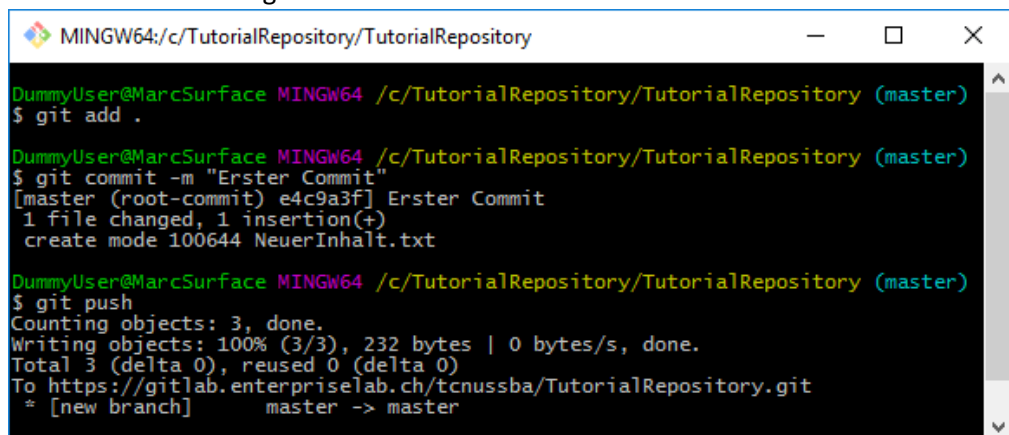
- a. Erstelle eine Text-Datei mit Namen gitignore.txt
- b. Kopiere folgenden Inhalt in diese Textdatei

```
[Ll]ibrary/  
[Tt]emp/  
[Oo]bj/  
[Bb]uild/  
[Bb]uilds/  
Assets/AssetStoreTools*  
  
# Visual Studio 2015 cache directory  
.vs/  
  
# Autogenerated VS/MD/Consulo solution and project files  
ExportedObj/  
.consulo/  
*.csproj  
*.unityproj  
*.sln  
*.suo  
*.tmp  
*.user  
*.userprefs  
*.pidb  
*.booproj  
*.svd  
*.pdb  
Physics2DSettings.asset  
DynamicsManager.asset  
  
# Unity3D generated meta files  
*.pidb.meta  
  
# Unity3D Generated File On Crash Reports  
sysinfo.txt  
  
# Builds  
*.apk  
*.unitypackage
```

- c. Benenne die Datei zu «**.gitignore**» um. Ja Punkt-gitignore-Punkt, dadurch wird die Datei zu «.gitignore». Ja Windows ist hier komisch.

5. Upload von Änderungen

- a. Im Ordner die Git-Bash öffnen und folgende Befehle eingeben, wobei «Erster Commit» eine beliebige Nachricht für den Commit sein kann



```
MINGW64:/c:/TutorialRepository/TutorialRepository  
DummyUser@MarcSurface MINGW64 /c:/TutorialRepository/TutorialRepository (master)  
$ git add .  
DummyUser@MarcSurface MINGW64 /c:/TutorialRepository/TutorialRepository (master)  
$ git commit -m "Erster Commit"  
[master (root-commit) e4c9a3f] Erster Commit  
1 file changed, 1 insertion(+)  
create mode 100644 NeuerInhalt.txt  
DummyUser@MarcSurface MINGW64 /c:/TutorialRepository/TutorialRepository (master)  
$ git push  
Counting objects: 3, done.  
Writing objects: 100% (3/3), 232 bytes | 0 bytes/s, done.  
Total 3 (delta 0), reused 0 (delta 0)  
To https://gitlab.enterpriselab.ch/tcnussba/TutorialRepository.git  
* [new branch] master -> master
```

Mögliche Stolpersteine

- Unity-Projekte benötigen viel Speicherplatz, es wird schnell an die Default-Grösse eines Gitlab-Repository gestossen. Dies wird durch die folgende Fehlermeldung angezeigt:
error: unpack failed: unable to create temporary object directory
To https://gitlab.enterpriselab.ch/tcnussba/pawi-vortex-tunnel.git
! [remote rejected] master -> master (unpacker error)
- Falls die .gitignore-Datei nicht verwendet wird so gibt es bei jedem Push eines anderen Benutzers ein Konflikt. Dies passiert weil Unity sehr viele temporäre lokale Dateien erzeugt.
- Sind die Editor-Settings nicht wie im Tutorial beschrieben gesetzt so ändert Unity Projekt-Assets was zu Konflikten führt beim Push/Pull.

Referenzen

<https://github.com/github/gitignore/blob/master/Unity.gitignore>

(02.12.2017)

<https://robots.thoughtbot.com/how-to-git-with-unity>

(02.12.2017)