

Erste Schritte mit Git ... in Verbindung mit Lazarus

Einleitung

Was macht man mit git eigentlich. Mit Git kann man verschiedene Versionen seines Programms speichern und auch (hoffentlich) wieder finden.

Wikipedia: Git [[git](#)] ist eine [freie Software](#) zur [verteilten Versionsverwaltung](#) von [Dateien](#), die durch [Linus Torvalds](#) initiiert wurde.

Meine Quellen:

<https://git-scm.com/book/de/v2/Erste-Schritte-Die-Kommandozeile>

<https://www.it-swarm.com/de/de/git/>

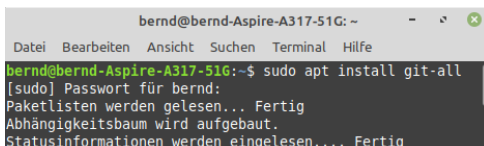
<https://www.atlassian.com/de/git/tutorials/what-is-version-control>

Wer wie ich mit eigentlich null Ahnung an dieses Thema ran geht dem sei zuerst gesagt git ist ein Terminal-Programm das lokal auf der eigenen Festplatte ausgeführt wird. Ob man später GitHub, GitLab oder sonst was nutzen möchte ist für den Anfang erst mal egal.

Ich habe zum Einstieg mehrere sehr gute Tutorials gelesen. Leider enthielten aber alle diese Tutorials für einen blutigen Neuling, wie mich, zu viele Informationen auf ein mal, so das mir oft schon nach kurzer Zeit die Lust verging weiter zu machen. Aus diesem Grund möchte ich hier meinen Weg, der alles weg lässt was am Anfang eh nur verwirrt, vorstellen. Ich verwende als OS Linux Mint, es sollte aber wohl mit anderen Betriebssystemen ziemlich ähnlich von statten gehen.

Git installieren

Ein Terminal öffnen und `sudo apt install git-all` eingeben.

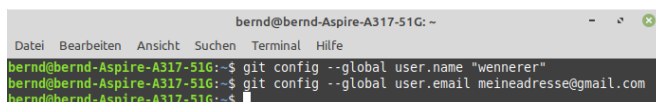


```
bernd@bernd-Aspire-A317-51G: ~  
Datei Bearbeiten Ansicht Suchen Terminal Hilfe  
bernd@bernd-Aspire-A317-51G:~$ sudo apt install git-all  
[sudo] Passwort für bernd:  
Paketlisten werden gelesen... Fertig  
Abhängigkeitsbaum wird aufgebaut.  
Statusinformationen werden eingelesen.... Fertig
```

Git konfigurieren

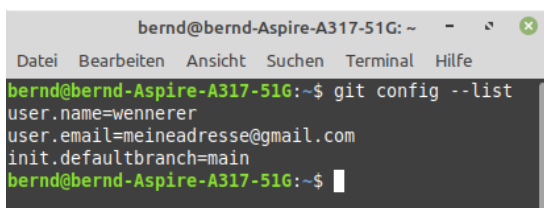
Den Benutzernamen festlegen : `git config --global user.name "DEIN BENUTZERNAME"`

E-mailadresse festlegen : `git config --global user.email deineadresse@gmail.com`



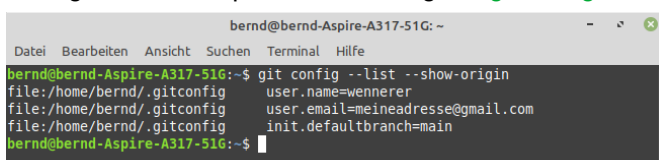
```
bernd@bernd-Aspire-A317-51G: ~  
Datei Bearbeiten Ansicht Suchen Terminal Hilfe  
bernd@bernd-Aspire-A317-51G:~$ git config --global user.name "wennerer"  
bernd@bernd-Aspire-A317-51G:~$ git config --global user.email meineadresse@gmail.com  
bernd@bernd-Aspire-A317-51G:~$
```

Konfiguration anzeigen : `git config --list`



```
bernd@bernd-Aspire-A317-51G: ~  
Datei Bearbeiten Ansicht Suchen Terminal Hilfe  
bernd@bernd-Aspire-A317-51G:~$ git config --list  
user.name=wennerer  
user.email=meineadresse@gmail.com  
init.defaultbranch=main  
bernd@bernd-Aspire-A317-51G:~$
```

Konfiguration mit Speicherort anzeigen : `git config --list --show-origin`



```
bernd@bernd-Aspire-A317-51G: ~  
Datei Bearbeiten Ansicht Suchen Terminal Hilfe  
bernd@bernd-Aspire-A317-51G:~$ git config --list --show-origin  
file:/home/bernd/.gitconfig user.name=wennerer  
file:/home/bernd/.gitconfig user.email=meineadresse@gmail.com  
file:/home/bernd/.gitconfig init.defaultbranch=main  
bernd@bernd-Aspire-A317-51G:~$
```

Optional wenn nicht der Standardeditor verwendet werden soll (ich wollte xed) folgendes im Terminal eingeben: `git config --global core.editor xed`

```
bernd@bernd-Aspire-A317-51G: ~  
Datei Bearbeiten Ansicht Suchen Terminal Hilfe  
bernd@bernd-Aspire-A317-51G:~$ git config --global core.editor xed  
bernd@bernd-Aspire-A317-51G:~$ git config --list  
user.name=wennerer  
user.email=meineadresse@gmail.com  
init.defaultbranch=main  
core.editor=xed  
bernd@bernd-Aspire-A317-51G:~$
```

Ein kleines Lazarus Programm zum Testen erzeugen

Hier ist der Quellcode meines Testprogramms:

```
unit unit_testgit;  
  
{$mode objfpc}{$H+}  
  
interface  
  
uses  
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls;  
  
type  
  
  { TForm1 }  
  
  TForm1 = class(TForm)  
    aLabel : TLabel;  
    procedure FormCreate(Sender: TObject);  
  private  
  
  public  
  
  end;  
  
var  
  Form1: TForm1;  
  
implementation  
  
{$R *.lfm}  
  
{ TForm1 }  
  
procedure TForm1.FormCreate(Sender: TObject);  
begin  
  caption      := 'Test Git';  
  setBounds(100,100,500,200);  
  
  aLabel       := TLabel.Create(self);  
  aLabel.Parent := self;  
  aLabel.AutoSize := false;  
  aLabel.SetBounds(150,90,200,20);  
  aLabel.Alignment := taCenter;  
  aLabel.Layout := tlCenter;  
  aLabel.Color := clWhite;  
  aLabel.Caption := 'This is the 1. Commit';  
end;  
  
end.
```

Die Ordnerstruktur eines normalen Lazarusprogrammverzeichnisses:

/home/bernd/Sprachen/Lazarus/64_bit/Git/Git01				
Name	Größe	Datentyp	Änderungsdatum	
backup	4 Objekte	Ordner	Mo 27 Dez 2021 16:29:57 CET	
TestGit.lpi	12,2 kB	Auszeichnung	Mo 27 Dez 2021 16:29:12 CET	
TestGit.lps	1,0 kB	Auszeichnung	Mo 27 Dez 2021 16:29:12 CET	
unit_testgit.lfm	149 Bytes	Text	Mo 27 Dez 2021 16:29:12 CET	
unit_testgit.pas	726 Bytes	Text	Mo 27 Dez 2021 16:29:12 CET	
lib	1 Objekt	Ordner	Mo 27 Dez 2021 16:29:19 CET	
x86_64-linux	7 Objekte	Ordner	Mo 27 Dez 2021 16:29:57 CET	
TestGit.co...	886 Bytes	Auszeichnung	Mo 27 Dez 2021 16:29:58 CET	
TestGit.o	40,7 kB	Dokument	Mo 27 Dez 2021 16:29:57 CET	
TestGit.or	207,8 kB	Dokument	Mo 27 Dez 2021 16:29:57 CET	
TestGit.res	139,1 kB	Binär	Mo 27 Dez 2021 16:29:57 CET	
unit_testg...	149 Bytes	Text	Mo 27 Dez 2021 16:29:57 CET	
unit_testg...	60,0 kB	Dokument	Mo 27 Dez 2021 16:29:57 CET	
unit_testg...	99,2 kB	Binär	Mo 27 Dez 2021 16:29:57 CET	
Linux_64	1 Objekt	Ordner	Mo 27 Dez 2021 16:29:58 CET	
TestGit	25,9 MB	Programm	Mo 27 Dez 2021 16:29:58 CET	
TestGit.ico	137,0 kB	Bild	Mo 27 Dez 2021 16:29:00 CET	
TestGit.lpi	12,2 kB	Auszeichnung	Mo 27 Dez 2021 16:29:57 CET	
TestGit.lpr	393 Bytes	Text	Mo 27 Dez 2021 16:29:12 CET	
TestGit.lps	1,0 kB	Auszeichnung	Mo 27 Dez 2021 16:29:57 CET	
TestGit.res	139,1 kB	Binär	Mo 27 Dez 2021 16:29:57 CET	
unit_testgit.lfm	149 Bytes	Text	Mo 27 Dez 2021 16:29:57 CET	
unit_testgit.pas	760 Bytes	Text	Mo 27 Dez 2021 16:29:57 CET	

Die ausführbare Datei liegt bei mir hier zum Beispiel im Ordner Linux_64. Das kann je nach Lazarus Erstellmodi und IDE Einstellungen variieren!

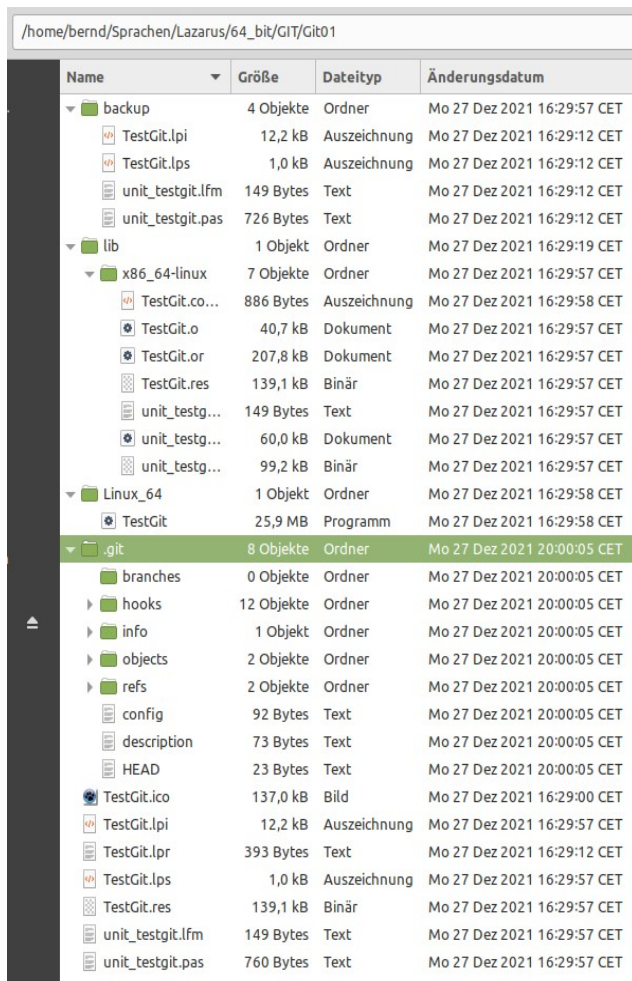
Ein lokales Git-Repository anlegen:

Das bedeutet das obige Ordnerstruktur auf Änderungen überwacht wird.

Ein Terminal öffnen und mit `cd` in obigen Ordner wechseln. Dann `git init` eingeben.

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git init
Leeres Git-Repository in /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01/.git/ initialisiert
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

Schaut man sich nun die Ordnerstruktur an und schaltet die verborgenen Dateien auf sichtbar (Strg+H) sieht es so aus:



Name	Größe	Dateityp	Änderungsdatum
backup	4 Objekte	Ordner	Mo 27 Dez 2021 16:29:57 CET
TestGit.lpi	12,2 kB	Auszeichnung	Mo 27 Dez 2021 16:29:12 CET
TestGit.lps	1,0 kB	Auszeichnung	Mo 27 Dez 2021 16:29:12 CET
unit_testgit.lfm	149 Bytes	Text	Mo 27 Dez 2021 16:29:12 CET
unit_testgit.pas	726 Bytes	Text	Mo 27 Dez 2021 16:29:12 CET
lib	1 Objekt	Ordner	Mo 27 Dez 2021 16:29:19 CET
x86_64-linux	7 Objekte	Ordner	Mo 27 Dez 2021 16:29:57 CET
TestGit.co...	886 Bytes	Auszeichnung	Mo 27 Dez 2021 16:29:58 CET
TestGit.o	40,7 kB	Dokument	Mo 27 Dez 2021 16:29:57 CET
TestGit.or	207,8 kB	Dokument	Mo 27 Dez 2021 16:29:57 CET
TestGit.res	139,1 kB	Binär	Mo 27 Dez 2021 16:29:57 CET
unit_testg...	149 Bytes	Text	Mo 27 Dez 2021 16:29:57 CET
unit_testg...	60,0 kB	Dokument	Mo 27 Dez 2021 16:29:57 CET
unit_testg...	99,2 kB	Binär	Mo 27 Dez 2021 16:29:57 CET
Linux_64	1 Objekt	Ordner	Mo 27 Dez 2021 16:29:58 CET
TestGit	25,9 MB	Programm	Mo 27 Dez 2021 16:29:58 CET
.git	8 Objekte	Ordner	Mo 27 Dez 2021 20:00:05 CET
branches	0 Objekte	Ordner	Mo 27 Dez 2021 20:00:05 CET
hooks	12 Objekte	Ordner	Mo 27 Dez 2021 20:00:05 CET
info	1 Objekt	Ordner	Mo 27 Dez 2021 20:00:05 CET
objects	2 Objekte	Ordner	Mo 27 Dez 2021 20:00:05 CET
refs	2 Objekte	Ordner	Mo 27 Dez 2021 20:00:05 CET
config	92 Bytes	Text	Mo 27 Dez 2021 20:00:05 CET
description	73 Bytes	Text	Mo 27 Dez 2021 20:00:05 CET
HEAD	23 Bytes	Text	Mo 27 Dez 2021 20:00:05 CET
TestGit.ico	137,0 kB	Bild	Mo 27 Dez 2021 16:29:00 CET
TestGit.lpi	12,2 kB	Auszeichnung	Mo 27 Dez 2021 16:29:57 CET
TestGit.lpr	393 Bytes	Text	Mo 27 Dez 2021 16:29:12 CET
TestGit.lps	1,0 kB	Auszeichnung	Mo 27 Dez 2021 16:29:57 CET
TestGit.res	139,1 kB	Binär	Mo 27 Dez 2021 16:29:57 CET
unit_testgit.lfm	149 Bytes	Text	Mo 27 Dez 2021 16:29:57 CET
unit_testgit.pas	760 Bytes	Text	Mo 27 Dez 2021 16:29:57 CET

Es wurde der Ordner `.git` mit etlichen Unterordnern und Dateien angelegt.

Um den Status des Repositories zu sehen kann man im Terminal `git status` eingeben.

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git status
Auf Branch master

Noch keine Commits

Unversionierte Dateien:
(benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
TestGit.lco
TestGit.lpi
TestGit.lpr
TestGit.lps
TestGit.res
backup/
lib/
unit_testgit.lfm
unit_testgit.pas

nichts zum Commit vorgemerkt, aber es gibt unversionierte Dateien
(benutzen Sie "git add" zum Versionieren)
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

Wir befinden uns auf den Branch „Master“ (Main) also auf dem Hauptzweig.

Es wurden noch keine Commits getätigt. Es wurden also noch keine Versionsstände gespeichert.

Alle roten Dateien und Verzeichnisse können mit `git add` zum Versionieren angemeldet werden. Das bedeutet das dann dieser Zustand beim nächsten Commit gesichert wird.

Dateien im lokalen Git-Repository ignorieren

In jedem Lazarus Projektordner befinden sich Ordner und Dateien, wie *backup*, die man wahrscheinlich nicht in die Versionsverwaltung aufnehmen möchte. Dies erreicht man in dem man eine Textdatei mit den Namen *.gitignore* erzeugt und in diese die Dateien und Verzeichnisse die ignoriert werden sollen hinein schreibt.

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ cat > .gitignore
backup
lib
Linux_64
.gitignore
^Z
[1]+  Angehalten                  cat > .gitignore
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

```
cat > .gitignore
backup
lib
Linux_64
.gitignore
```

mit Strg+Z verlassen.

Oder einfach einen Texteditor öffnen und die Datei erzeugen!

/home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01

Name	Größe	Dateityp	Änderungsdatum
backup	4 Objekte	Ordner	Mo 27 Dez 2021 16:29:57 CET
lib	1 Objekt	Ordner	Mo 27 Dez 2021 16:29:19 CET
Linux_64	1 Objekt	Ordner	Mo 27 Dez 2021 16:29:58 CET
.git	8 Objekte	Ordner	Mo 27 Dez 2021 20:22:30 CET
TestGit.ico	137,0 kB	Bild	Mo 27 Dez 2021 16:29:00 CET
TestGit.lpi	12,2 kB	Auszeichnung	Mo 27 Dez 2021 16:29:57 CET
TestGit.lpr	393 Bytes	Text	Mo 27 Dez 2021 16:29:12 CET
TestGit.lps	1,0 kB	Auszeichnung	Mo 27 Dez 2021 16:29:57 CET
TestGit.res	139,1 kB	Binär	Mo 27 Dez 2021 16:29:57 CET
unit_testgit.lfm	149 Bytes	Text	Mo 27 Dez 2021 16:29:57 CET
unit_testgit.pas	760 Bytes	Text	Mo 27 Dez 2021 16:29:57 CET
.gitignore	31 Bytes	Text	Mo 27 Dez 2021 20:46:01 CET

Datei Bearbeiten Ansicht

.gitignore x

```
backup
lib
Linux_64
.gitignore
```

Nun nochmal im Terminal *git status* eingeben

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git status
Auf Branch master

Noch keine Commits

Unversionierte Dateien:
  (benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
    TestGit.ico
    TestGit.lpi
    TestGit.lpr
    TestGit.lps
    TestGit.res
    unit_testgit.lfm
    unit_testgit.pas

nichts zum Commit vorgemerkt, aber es gibt unversionierte Dateien
(benutzen Sie "git add" zum Versionieren)
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

Die in der Datei *.gitignore* eingetragenen Dateien und Verzeichnisse werden nicht mehr angezeigt.

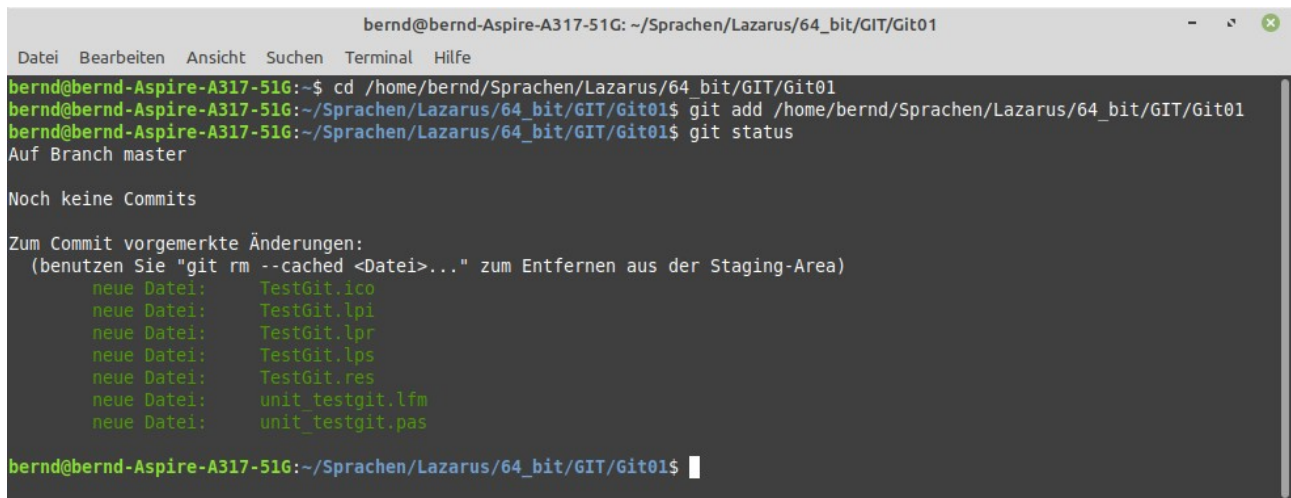
Ignorieren einer bereits committeten Datei

```
git rm --cached TestProjekte/TestProjekt_MP_3/project1.dbg
```

Dann noch *project1.dbg* (oder **.dbg*) in *.gitignore* aufnehmen

Dateien im lokalen Git-Repository zum Versionieren hinzufügen

Mit `git add` können nun Dateien und Verzeichnisse für den nächsten Commit vorgemerkt werden. Da wir mit `.gitignore` bereits die zu ignorierenden Verzeichnisse deklariert haben bietet es sich hier an das ganze Verzeichnis `Git01` für einen ersten Commit hinzu zufügen. Alternativ steht der Befehl `git add -all` zur Verfügung um alle Dateien auf einmal vorzumerken.



```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git add /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git status
Auf Branch master

Noch keine Commits

Zum Commit vorgemerkte Änderungen:
(benutzen Sie "git rm --cached <Datei>..." zum Entfernen aus der Staging-Area)
    neue Datei:    TestGit.ico
    neue Datei:    TestGit.lpi
    neue Datei:    TestGit.lpr
    neue Datei:    TestGit.lps
    neue Datei:    TestGit.res
    neue Datei:    unit_testgit.lfm
    neue Datei:    unit_testgit.pas

bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

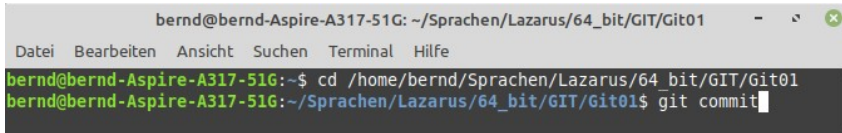
Möchte man jetzt noch eine Datei wieder entfernen so könnte man dies z. Bsp. mit `git rm --cached TestGit.ico` tun.

Einen Commit durchführen (sichern)

Immer wenn ein Meilenstein auf dem Weg zum fertigen Programm erreicht ist kann mit einem Commit eine Sicherung gemacht werden. Es ist dann später möglich zu diesem Stand zurück zu kehren.

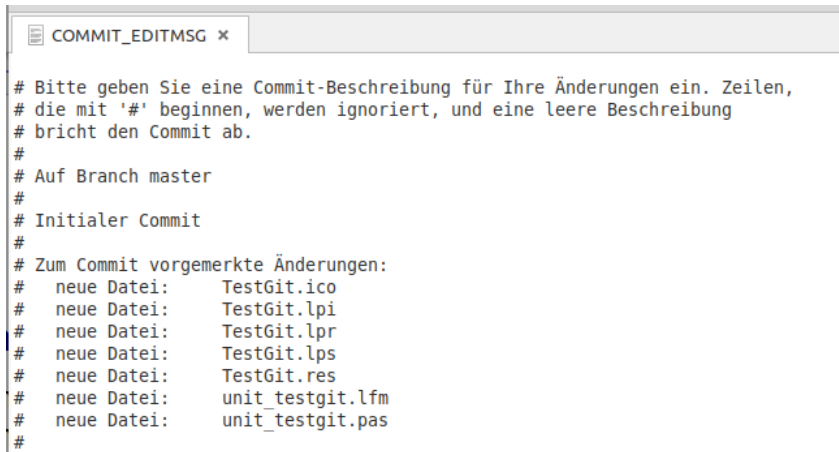
Wir sichern nun den Zustand der oben hinzugefügten Dateien.

Dazu im Terminal **git commit** eingeben.



```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git commit
```

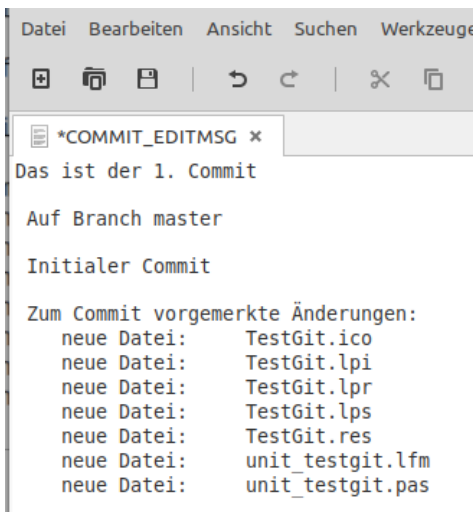
Nach dem Enter gedrückt wurde öffnet sich der Texteditor (bei mir xed):



```
COMMIT_EDITMSG
# Bitte geben Sie eine Commit-Beschreibung für Ihre Änderungen ein. Zeilen,
# die mit '#' beginnen, werden ignoriert, und eine leere Beschreibung
# bricht den Commit ab.
#
# Auf Branch master
#
# Initialer Commit
#
# Zum Commit vorgemerkte Änderungen:
#   neue Datei:    TestGit.ico
#   neue Datei:    TestGit.lpi
#   neue Datei:    TestGit.lpr
#   neue Datei:    TestGit.lps
#   neue Datei:    TestGit.res
#   neue Datei:    unit_testgit.lfm
#   neue Datei:    unit_testgit.pas
#
```

Hier kann bzw. muss eingetragen werden was man später zum identifizieren des Meilensteins braucht. Alle Zeilen die mit # beginnen werden ignoriert und nicht in den Commit geschrieben. Gibt es keine Zeile die in den Commit geschrieben wird, wird der Commit abgebrochen! Tipp: mit Suchen und Ersetzen kann man alle # entfernen.

Kann dann so aussehen:



```
*COMMIT_EDITMSG
Das ist der 1. Commit

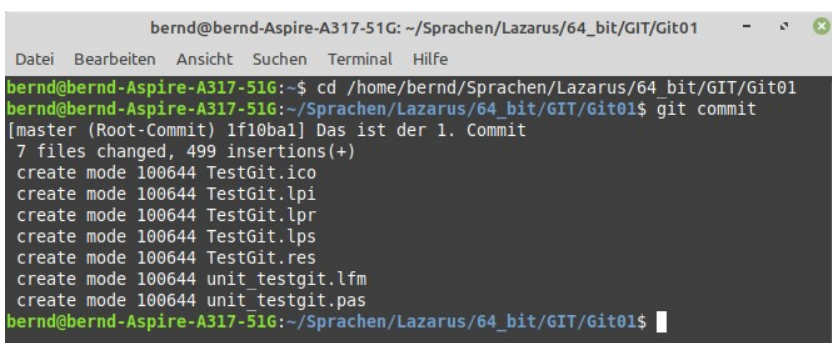
Auf Branch master

Initialer Commit

Zum Commit vorgemerkte Änderungen:
  neue Datei:    TestGit.ico
  neue Datei:    TestGit.lpi
  neue Datei:    TestGit.lpr
  neue Datei:    TestGit.lps
  neue Datei:    TestGit.res
  neue Datei:    unit_testgit.lfm
  neue Datei:    unit_testgit.pas
```

Datei speichern und schließen.

Im Terminal schaut es jetzt so aus:



```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git commit
[master (Root-Commit) 1f10ba1] Das ist der 1. Commit
7 files changed, 499 insertions(+)
create mode 100644 TestGit.ico
create mode 100644 TestGit.lpi
create mode 100644 TestGit.lpr
create mode 100644 TestGit.lps
create mode 100644 TestGit.res
create mode 100644 unit_testgit.lfm
create mode 100644 unit_testgit.pas
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

git status ergibt folgende Ausgabe:

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git status
Auf Branch master
nichts zu committen, Arbeitsverzeichnis unverändert
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

Nun verändern wir im Quellcode die Caption des Labels in `aLabel.Caption` := 'This is the 2. Commit'; und speichern (oder kompilieren) das Programm in der Lazarus IDE.

git status ergibt folgende Ausgabe:

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git status
Auf Branch master
Änderungen, die nicht zum Commit vorgemerkt sind:
  (benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
  (benutzen Sie "git restore <Datei>...", um die Änderungen im Arbeitsverzeichnis zu verwerfen)
    geändert:      unit_testgit.pas

keine Änderungen zum Commit vorgemerkt (benutzen Sie "git add" und/oder "git commit -a")
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

Als nächstes werden wir die geänderte Datei `unit_testgit.pas` wieder mit `git add` vormerken und dann den 2. Commit mit der Option `-v` durchführen. Die Option bewirkt dass die Änderungen mit angezeigt werden.

```
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git add unit_testgit.pas

bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git commit -v
[master c79a9f5] Das ist der 2. Commit
 1 file changed, 1 insertion(+), 1 deletion(-)
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

Das wiederholen wir nun bis wir 5 Commits haben. Bitte beachten kompiliert man das Programm werden eventuell mehrere Dateien geändert die committet werden sollten.

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git status
Auf Branch master
Änderungen, die nicht zum Commit vorgemerkt sind:
  (benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
  (benutzen Sie "git restore <Datei>...", um die Änderungen im Arbeitsverzeichnis zu verwerfen)
    geändert:      TestGit.lpi
    geändert:      TestGit.lps
    geändert:      unit_testgit.pas

keine Änderungen zum Commit vorgemerkt (benutzen Sie "git add" und/oder "git commit -a")
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git add TestGit.lpi TestGit.lps unit_testgit.pas
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git commit
```

Alles auf einmal: `git commit -a -m 'Kommentar'`

Anzeigen aller Commits

Mit dem Befehl `git log` kann man sich alle gemachten Commits (Sicherungen) ansehen. Passen nicht alle Commits in das Terminalfenster sollte man an den Befehl noch `| more` anhängen um durch das Terminal blättern zu können. Also `git log | more`

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git log | more
commit ff7909225e76dfb9b7ce654d37cd35c1aa8af1bf
Author: wannerer <meineadresse@gmail.com>
Date: Mon Dec 27 22:34:28 2021 +0100

    Das ist der 5. Commit

    Auf Branch master
    Zum Commit vorgemerkte Änderungen:
        geändert: unit_testgit.pas

commit 291aa17467d888822616bdf0b37da61f229b390d
Author: wannerer <meineadresse@gmail.com>
Date: Mon Dec 27 22:32:22 2021 +0100

    Das ist der 4. Commit
    - aLabel.Caption      := 'This is the 3. Commit';
    + aLabel.Caption      := 'This is the 4. Commit';

    Auf Branch master
    Zum Commit vorgemerkte Änderungen:
        geändert: unit_testgit.pas

    ----- >8 -----
    Ändern oder entfernen Sie nicht die obige Zeile.
    Alles unterhalb von ihr wird ignoriert.
    diff --git a/unit_testgit.pas b/unit_testgit.pas
    index 364fe23..709e3de 100644
    --- a/unit_testgit.pas
    +++ b/unit_testgit.pas
    @@ -41,7 +41,7 @@ begin
        aLabel.Alignment      := taCenter;
        aLabel.Layout         := tlCenter;
        aLabel.Color          := clWhite;
    - aLabel.Caption          := 'This is the 3. Commit';
    + aLabel.Caption          := 'This is the 4. Commit';
    end;

    end.

commit 9f5b7fecffcabd6b14d860dd7e9aaf221c43868f
Author: wannerer <meineadresse@gmail.com>
Date: Mon Dec 27 22:29:47 2021 +0100

    Das ist der 3. Commit

    Auf Branch master
    Zum Commit vorgemerkte Änderungen:
        geändert: TestGit.lpi
        geändert: TestGit.lps
        geändert: unit_testgit.pas

--Mehr--
```

Nachträglich den letzten Kommentar ändern

Dazu `git commit --amend` ins Terminal eingeben. Es öffnet sich das letzte Kommentar im Texteditor.

Änderungen Rückgängig machen die noch nicht committed sind

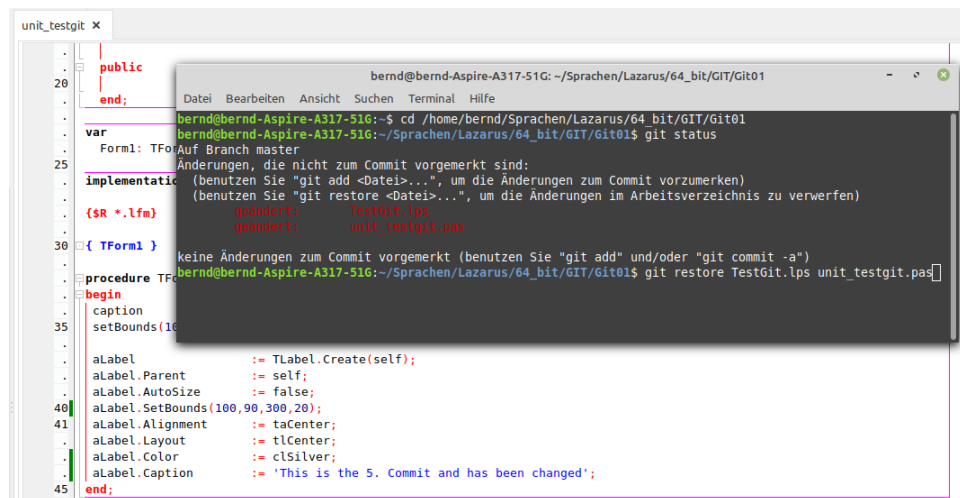
Dies geschieht mit dem Befehl `git restore`. Achtung da die Änderungen noch nicht gesichert wurden sind diese danach wirklich weg!

Ändern Sie den Quellcode im Lazarus Programm folgendermaßen ab :

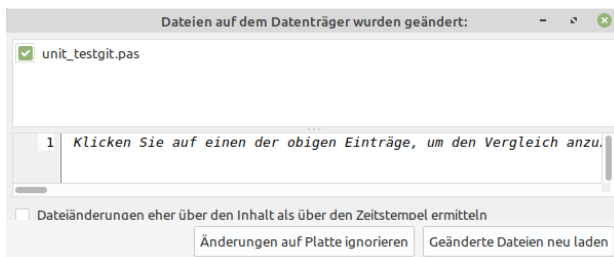
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  caption      := 'Test Git';
  setBounds(100,100,500,200);

  aLabel       := TLabel.Create(self);
  aLabel.Parent := self;
  aLabel.AutoSize := false;
  aLabel.SetBounds(100,90,300,20);
  aLabel.Alignment := taCenter;
  aLabel.Layout := tlCenter;
  aLabel.Color := clSilver;
  aLabel.Caption := 'This is the 5. Commit and has been changed';
end;
```

Mit `git status` sieht man nun das Änderungen vorgenommen wurden. Um den Zustand des letzten Commits herzustellen gibt man nun `git restore TestGit.lps unit_testgit.pas` ein.



Klickt man nun in den Lazarus Quelltext Editor erscheint dieses Fenster:



Mit geänderte Dateien neu laden wird der Zustand des letzten Commits wieder hergestellt.

Tipp: Mit `git restore '*'` setzt man alle Dateien auf einmal zurück!

Zu einen Commit zurückspringen und nachfolgende Commits entfernen

Dafür gibt es mehrere Methoden. Hier auf der lokalen Festplatte möchte ich `git reset --hard Id && git clean -f` verwenden. *Achtung: Commits zu entfernen ist nur ratsam wenn man alleine an einem Projekt arbeitet. Die entfernten Commits sind endgültig weg!*
Um zu dem 3. Commit zurück zukehren zuerst mit `git log | more` die ID des 3. Commits besorgen.

```
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git log | more

commit 9f5b7fecffcabd6b14d860dd7e9aaf221c43868f
Author: wennerer <meineadresse@gmail.com>
Date:   Mon Dec 27 22:29:47 2021 +0100

    Das ist der 3. Commit

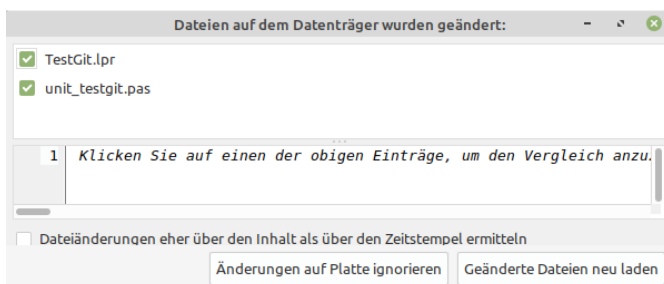
    Auf Branch master
    Zum Commit vorgemerkte Änderungen:
        geändert:   TestGit.lpi
        geändert:   TestGit.lps
        geändert:   unit_testgit.pas
```

Wie man hier sieht ist die Id für den 3. Commit
9f5b7fecffcabd6b14d860dd7e9aaf221c43868f

Jetzt `git reset --hard 9f5b7fecffcabd6b14d860dd7e9aaf221c43868f && git clean -f` eingeben.

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git reset --hard 9f5b7fecffcabd6b14d860dd7e9aaf221c43868f && git clean -f
HEAD ist jetzt bei 9f5b7fe Das ist der 3. Commit
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

In der Lazarus IDE erscheint nun folgendes Fenster



Nachdem „Geänderte Dateien neu laden“ gedrückt wurde ist der Quelltext im Zustand des 3. Commits.

Hinweis: 1x musste ich beim Testen Lazarus schließen und neu öffnen damit sich der Quelltext aktualisierte. Warum auch immer?

In einen früheren Commit springen, testen und zurück kehren

Möchte man in einen früheren Commit springen um etwas aus zu testen geht dies mit dem Befehl

git checkout

Ich möchte nun in den 2. Commit springen und dann wieder zurück in den 3.

Als erstes mit **git log | more** die ID des 2. Commits ermitteln.

```
commit c79a9f59d35413c5036996c3dd515e45f170806c
Author: wennerer <meineadresse@gmail.com>
Date:   Mon Dec 27 22:19:50 2021 +0100
```

Das ist der 2. Commit

Auf Branch master

Zum Commit vorgemerkte Änderungen:
geändert: unit_testgit.pas

Die Id ist hier

c79a9f59d35413c5036996c3dd515e45f170806c

Jetzt **git checkout c79a9f59d35413c5036996c3dd515e45f170806c** eingeben.

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git checkout c79a9f59d35413c5036996c3dd515e45f170806c
Hinweis: Wechsle zu 'c79a9f59d35413c5036996c3dd515e45f170806c'.

Sie befinden sich im Zustand eines 'losgelösten HEAD'. Sie können sich
umschauen, experimentelle Änderungen vornehmen und diese committen, und
Sie können alle möglichen Commits, die Sie in diesem Zustand machen,
ohne Auswirkungen auf irgendeinen Branch zu werfen, indem Sie zu einem
anderen Branch wechseln.

Wenn Sie einen neuen Branch erstellen möchten, um Ihre erstellten Commits
zu behalten, können Sie das (jetzt oder später) durch Nutzung von
'switch' mit der Option -c tun. Beispiel:

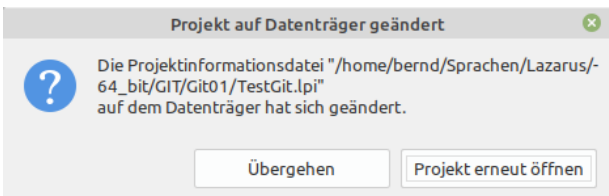
git switch -c <neuer-Branchname>

Oder um diese Operation rückgängig zu machen:
git switch -

Sie können diesen Hinweis ausschalten, indem Sie die Konfigurationsvariable
'advice.detachedHead' auf 'false' setzen.

HEAD ist jetzt bei c79a9f5 Das ist der 2. Commit
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

In der Lazarus IDE erscheint folgende Meldung:



Nachdem „Projekt erneut öffnen“ gedrückt wurde ist der Quelltext im Zustand des 2. Commits.

Nun z. Bsp. folgende Änderung vornehmen: `aLabel.Color := clSilver;` und neu kompilieren.

Um nun wieder in den 3. Commit zu wechseln ohne den 2. zu ändern folgendes in Terminal machen:

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git status
HEAD losgelöst bei c79a9f5
Änderungen, die nicht zum Commit vorgemerkt sind:
  (benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
  (benutzen Sie "git restore <Datei>...", um die Änderungen im Arbeitsverzeichnis zu werfen)
    geändert:   TestGit.lpi
    geändert:   TestGit.lps
    geändert:   unit_testgit.pas

keine Änderungen zum Commit vorgemerkt (benutzen Sie "git add" und/oder "git commit -a")
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git restore TestGit.lpi TestGit.lps unit_testgit.pas
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git checkout master
Vorherige Position von HEAD war c79a9f5 Das ist der 2. Commit
Zu Zweig »master« gewechselt
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

In der Lazarus IDE die geänderte Datei noch neu laden.

In einen früheren Commit springen, ändern und neu commiten

Möchte man in einen früheren Commit springen um von dort aus neu zu entwickeln geht dies mit dem Befehl **git checkout**

Ich möchte nun in den 2. Commit springen und dann einen neuen Commit 2.1 in einem neuen Zweig machen. Der 3. Commit bleibt dabei unberührt.

Als erstes mit **git log | more** die ID des 2. Commits ermitteln.

```
commit c79a9f59d35413c5036996c3dd515e45f170806c
Author: wennerer <meineadresse@gmail.com>
Date:   Mon Dec 27 22:19:50 2021 +0100

    Das ist der 2. Commit

    Auf Branch master
    Zum Commit vorgemerkte Änderungen:
        geändert:    unit_testgit.pas
```

Die Id ist hier

c79a9f59d35413c5036996c3dd515e45f170806c

Jetzt **git checkout c79a9f59d35413c5036996c3dd515e45f170806c** eingeben.

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/Git01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/Git01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git checkout c79a9f59d35413c5036996c3dd515e45f170806c
Hinweis: Wechsle zu 'c79a9f59d35413c5036996c3dd515e45f170806c'.

Sie befinden sich im Zustand eines 'losgelösten HEAD'. Sie können sich
umschauen, experimentelle Änderungen vornehmen und diese committen, und
Sie können alle möglichen Commits, die Sie in diesem Zustand machen,
ohne Auswirkungen auf irgendeinen Branch werfen, indem Sie zu einem
anderen Branch wechseln.

Wenn Sie einen neuen Branch erstellen möchten, um Ihre erstellten Commits
zu behalten, können Sie das (jetzt oder später) durch Nutzung von
'switch' mit der Option -c tun. Beispiel:

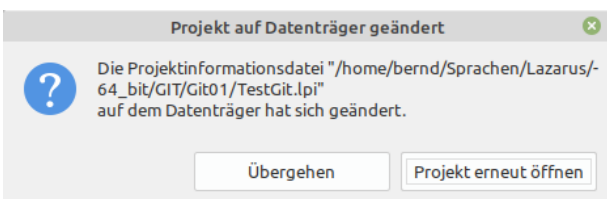
    git switch -c <neuer-Branchname>

Oder um diese Operation rückgängig zu machen:
    git switch -

Sie können diesen Hinweis ausschalten, indem Sie die Konfigurationsvariable
'advice.detachedHead' auf 'false' setzen.

HEAD ist jetzt bei c79a9f5 Das ist der 2. Commit
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

In der Lazarus IDE erscheint folgende Meldung:



Nachdem „Projekt erneut öffnen“ gedrückt wurde ist der Quelltext im Zustand des 2. Commits.

Nun z. Bsp. folgende Änderung vornehmen:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    caption      := 'Test Git';
    setBounds(100,100,500,200);

    aLabel       := TLabel.Create(self);
    aLabel.Parent := self;
    aLabel.AutoSize := false;
    aLabel.SetBounds(125,90,250,20);
    aLabel.Alignment := taCenter;
    aLabel.Layout    := tlCenter;
    aLabel.Color     := clSilver;
    aLabel.Caption   := 'This is the 2.1 Commit on a new branch';
end;
```

Das geänderte Programm in Lazarus kompilieren.

Einen neuen Branch (Zweig) erstellen

Um das geänderte Programm in einem neuen Zweig zu sichern folgendes ins Terminal eingeben:

`git switch -c NewBranch`

NewBranch steht hier für den Namen des neuen Zweiges und kann frei vergeben werden.

```
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git switch -c NewBranch
Zu neuem Branch 'NewBranch' gewechselt
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git status
Auf Branch NewBranch
Änderungen, die nicht zum Commit vorgemerkt sind:
  (benutzen Sie "git add <Datei>...", um die Änderungen zum Commit vorzumerken)
  (benutzen Sie "git restore <Datei>...", um die Änderungen im Arbeitsverzeichnis zu verworfen)
   geändert:    TestGit.lpi
   geändert:    TestGit.lps
   geändert:    unit_testgit.pas

keine Änderungen zum Commit vorgemerkt (benutzen Sie "git add" und/oder "git commit -a")
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$
```

Jetzt kann man die geänderten Dateien mit `git commit -a` gleichzeitig vormerken und commiten.

```
COMMIT_EDITMSG *
Das ist der 2.1 Commit auf einem neuen Zweig!

Auf Branch NewBranch
Zum Commit vorgemerkte Änderungen:
   geändert:    TestGit.lpi
   geändert:    TestGit.lps
   geändert:    unit_testgit.pas

bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git commit -a
[NewBranch 0692229] Das ist der 2.1 Commit auf einem neuen Zweig!
3 files changed, 7 insertions(+), 5 deletions(-)
```

Mit `git branch` werden alle Zweige angezeigt:

```
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git branch
* NewBranch
master
```

Mit `git branch -v` wird der jeweils letzte commit jeden Zweigs angezeigt:

```
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git branch -v
* NewBranch 0692229 Das ist der 2.1 Commit auf einem neuen Zweig!
master      9f5b7fe Das ist der 3. Commit
```


Den Branch (Zweig) wechseln

Mit `git checkout master` wechselt man in den letzten Commit des Master (**main**) Zweiges.

```
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git checkout master  
Zu Zweig »master« gewechselt
```

In der Lazarus IDE die geänderte Datei noch neu laden.

Mit `git checkout NewBranch` wechselt man in den letzten Commit des NewBranch Zweiges.

```
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/Git01$ git checkout NewBranch  
Zu Zweig »NewBranch« gewechselt
```

In der Lazarus IDE die geänderte Datei noch neu laden.

Ein neues Projekt nach GitHub pushen

Das Anmelden und Einrichten habe ich nach dieser Anleitung gemacht:

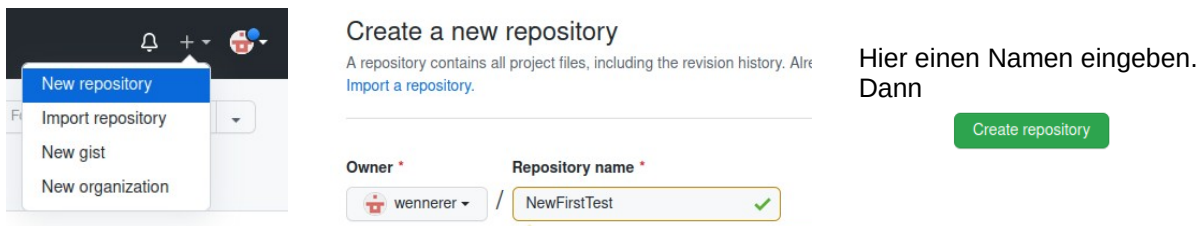
<https://git-scm.com/book/de/v2/GitHub-Einrichten-und-Konfigurieren-eines-Kontos>

Man benötigt zum Übertragen der Dateien nach GitHub aus Sicherheitsgründen SSH. Man kann sich im Terminal nicht mit dem Passwort einloggen!

Wie man einen SSH Schlüssel generiert steht hier:

https://git-scm.com/book/de/v2/Git-auf-dem-Server-Erstellung-eines-SSH-Public-Keys#_generate_ssh_key

Um sein Projekt auf GitHub zu übertragen erstellt man in GitHub zuerst ein neues Repository



Es erscheint dieser Bildschirm: Falls nicht angewählt auf SSH drücken!

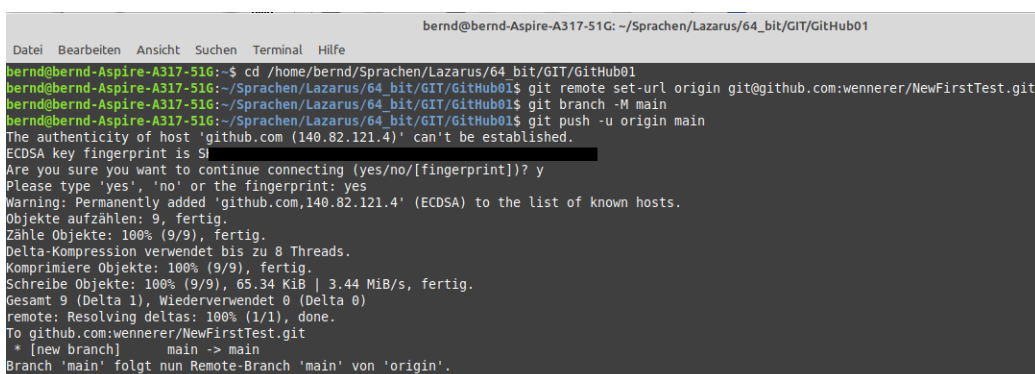


Im Prinzip muss man die Befehle im roten Rahmen hernehmen. Damit es funktioniert muss man die Zeile mit den Lila-Rahmen aber nur beim Ersten Commit eingeben. Zwischen der 2. und der 3. Zeile wird dieser Befehl benötigt:

`git remote set-url origin git@github.com:wennerer/NewFirstTest.git`

Ansonsten wird man im Terminal nach dem Passwort gefragt (die Passwort-Authentifizierung ist nicht mehr möglich).

```
cd .....
git remote add origin git@github.com:wennerer/NewFirstTest.git
git remote set-url origin git@github.com:wennerer/NewFirstTest.git
git branch -M main
git push -u origin main
```



Änderungen eines bestehenden Projektes nach GitHub pushen

Zuerst lokal das geänderte Projekt commiten.

mit `cd` ins Verzeichnis wechseln

`git commit -a -m BESCHREIBUNG`

dann nach github pushen

`git remote set-url origin git@github.com:wennerer/NewFirstTest.git`

`git branch -M main`

`git push -u origin main`

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/GIT/GitHub01
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/GIT/GitHub01
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/GitHub01$ git commit -a -m 4.Commit
[main bb5d90b] 4.Commit
 4 files changed, 4 insertions(+), 1 deletion(-)
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/GitHub01$ git remote set-url origin git@github.com:wennerer/NewFirstTest.git
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/GitHub01$ git branch -M main
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/GitHub01$ git push -u origin main
Objekte aufzählen: 11, fertig.
Zähle Objekte: 100% (11/11), fertig.
Delta-Kompression verwendet bis zu 8 Threads.
Komprimiere Objekte: 100% (6/6), fertig.
Schreibe Objekte: 100% (6/6), 560 Bytes | 560.00 KiB/s, fertig.
Gesamt 6 (Delta 5), Wiederverwendet 0 (Delta 0)
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To github.com:wennerer/NewFirstTest.git
bb5d90b..bb5d90b main -> main
Branch 'main' folgt nun Remote-Branch 'main' von 'origin'.
```

main 1 branch 0 tags Go to file Add file Code

wennerer 4.Commit		bb5d90b 5 minutes ago 4 commits
project1.ico	Das ist ein Test für GitHub	5 hours ago
project1.lpi	4.Commit	5 minutes ago
project1.lpr	Das ist ein Test für GitHub	5 hours ago
project1.lps	4.Commit	5 minutes ago
project1.res	Das ist ein Test für GitHub	5 hours ago
unit1.lfm	4.Commit	5 minutes ago
unit1.pas	4.Commit	5 minutes ago

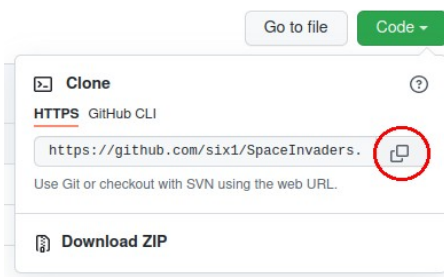
```
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/GIT/GitHub01$ git status
Auf Branch main
Ihr Branch ist auf demselben Stand wie 'origin/main'.

nichts zu committen, Arbeitsverzeichnis unverändert
```

Sich ein Projekt von GitHub holen

In meinem Beispiel klonen ich das Projekt „Space Invaders“ von Six1 (man gönnt sich ja sonst nichts).
siehe <https://www.lazarusforum.de/viewtopic.php?p=119973#p119973>

Als erstes muss man in den Ordner wechseln wohin man das Projekt kopieren möchte.
Anschließend wechselt man im Browser zu dem GitHub Projekt und kopiert sich dort die Adresse:



Dann gibt man im Terminal `git clone` und die kopierte Adresse ein.

```
bernd@bernd-Aspire-A317-51G: ~/Sprachen/Lazarus/64_bit/Spiele
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/Spiele
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/Spiele$ git clone https://github.com/six1/SpaceInvaders.git
Klone nach 'SpaceInvaders' ...
remote: Enumerating objects: 93, done.
remote: Counting objects: 100% (93/93), done.
remote: Compressing objects: 100% (86/86), done.
remote: Total 93 (delta 12), reused 0 (delta 0), pack-reused 0
Entpacke Objekte: 100% (93/93), 15.08 MiB | 9.95 MiB/s, fertig.
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/Spiele$
```

Das war es schon, jetzt befindet sich alles im Ordner Spiele!

Wenn das Git Hub Repository verändert wurde (Readme Datei)

Hat man im Repository was verändert benötigt man den Befehl `git pull` um das Git Hub mit dem lokalen Repository abzugleichen. Sonst bekommt man eine Fehlermeldung beim pushen!

Aktualisierungen wurden zurückgewiesen, weil das Remote-Repository Commits enthält die lokal nicht vorhanden sind.

`git remote set-url origin git@github.com:wennerer/Multis.git`
`git pull origin main`
`git push -u origin main`

```
bernd@bernd-Aspire-A317-51G:~$ cd /home/bernd/Sprachen/Lazarus/64_bit/Meine_Packages/Multi/Multis
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/Meine_Packages/Multi/Multis$ git status
Auf Branch main
Ihr Branch ist 8 Commits vor 'origin/main'.
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/Meine_Packages/Multi/Multis$ git remote set-url origin git@github.com:wennerer/Multis.git
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/Meine_Packages/Multi/Multis$ git pull origin master
fatal: Konnte Remote-Referenz master nicht finden.
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/Meine_Packages/Multi/Multis$ git pull origin main
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 24 (delta 14), reused 0 (delta 0), pack-reused 0
Entpacke Objekte: 100% (24/24), 5.60 KiB | 409.00 KiB/s, fertig.
Von github.com:wennerer/Multis
* branch      main      -> FETCH_HEAD
   99eff01..2b16ffb  main      -> origin/main
Merge made by the 'recursive' strategy.
 README.md | 10 ++++++++
 1 file changed, 10 insertions(+)
 create mode 100644 README.md
bernd@bernd-Aspire-A317-51G:~/Sprachen/Lazarus/64_bit/Meine_Packages/Multi/Multis$ git push -u origin main
Objekte aufzählen: 82, fertig.
Zähle Objekte: 100% (72/72), fertig.
Delta-Kompression verwendet bis zu 8 Threads.
Komprimiere Objekte: 100% (60/60), fertig.
Schreibe Objekte: 100% (60/60), 22.48 KiB | 7.49 MiB/s, fertig.
Gesamt 60 (Delta 40), Wiederverwendet 0 (Delta 0)
remote: Resolving deltas: 100% (40/40), completed with 9 local objects.
To github.com:wennerer/Multis.git
   2b16ffb..d69ed74  main -> main
Branch 'main' folgt nun Remote-Branch 'main' von 'origin'.
```

Einen Branch mit dem Main-Branch vereinigen (mergen)

Wurde der Main-Branch nicht verändert lässt sich ein neuer Zweig ganz einfach mergen. Zuerst im New-Branch git status ausführen und sicherstellen das alles committed wurde. Dann in den Main-Branch wechseln und dort mergen.

`git status` (im New-Branch)

`git checkout main`

`git merge New-Branch` (New-Branch steht für den Namen des Zweigs)

Will man New-Branch löschen geht das mit dem Befehl:

`git branch -d New-Branch` (New-Branch steht für den Namen des Zweigs)