

# Git Annex: Dateien Synchronisieren, Sichern und Verwalten für Poweruser

---

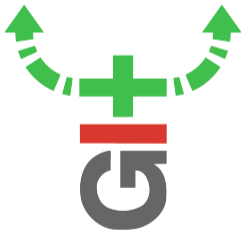
**Yann Büchau**

Uni Tübingen, Umweltphysik

Tübix 2023 – Tübingen

Workshops – 10:00-11:50 – W1/C215

01.07.2023 



 [git-annex.branchable.com](https://git-annex.branchable.com)

Woher kommt Git Annex?

Was kann Git Annex?

Was ist Git Annex?

Git Annex Workflow für einfachen, geteilten Ordner

Git Annex Assistant

Weiteres Material



 lwn.net

- ▶ **Joey Hess**
  - ▶ Langjähriger Debian Entwickler
  - ▶ Lebt in Holzhütte mit Solarstrom, Quellwasser und langsamem Internet
  - ▶ Brauchte einen Weg, Dateien gebündelt zu übertragen, wenn er mal schnelles Internet hat, z.B. in der Stadt
  - ▶ Brauchte einen Überblick, welche Dateien er wo schon gespeichert hatte, um unnötige, langsame Downloads zu vermeiden
- ⇒ Git Annex entwickelt

- ▶ Ordner synchronisieren wie Nextcloud Files, Syncthing, Dropbox, etc.
- ▶ Große/Binäre/sensible Dateien in ein git-Repository aufnehmen
- ▶ Den pull/commit/push-Nerv aus dem git-Workflow nehmen (einfach `git annex assist`)
- ▶ Komplette Änderungshistorie beliebiger Dateien behalten
- ▶ Redundante Backups von Dateien durchführen und im Überblick behalten
- ▶ Kurz mal Dateien löschen, die sicher woanders gespeichert sind
- ▶ Forschungsdaten katalogisieren/synchronisieren/sichern
- ▶ Eine Mediensammlung katalogisieren/synchronisieren/sichern
- ▶ funktioniert unter Linux/Mac/Windows, sogar Android (Termux)
- ▶ ...

- ▶ Ab etwa **100000 Dateien** im Repository wirds langsam
- ▶ Auf **langsamen USB-Sticks** und schlechten Dateisystemen (z.B. FAT) macht es keinen Spaß
- ▶ Es fehlen einfache **grafische Programme**
- ▶ Git Annex übernimmt *nicht* die **Verbindung von Repositories** für einen. Man muss selber Verbindungen zwischen PCs etc. herstellen (im Gegensatz zu Syncthing z.B.)
- ▶ Git Annex hat **keine Rechteverwaltung**. Jeder, der das Repository und push-Recht hat, darf alles tun.
- ▶ **Lernkurve**
- ▶ ...?

- ▶ **Erweiterung** zu Versionsverwaltung git
- ▶ git ist **toll** für Textdateien (Code, etc.)
- ▶ git ist **schlecht** für alles andere (Fotos, Videos, Musik, Daten, Archive, Spreadsheets, etc.), vor allem, wenn diese sich ändern
  - ▶ git speichert bei Binärdateien den **kompletten Inhalt jeder Version** quasi-unlöschar in der Versionsgeschichte ab
  - ▶ Das Repository wird riesig und langsam
- ▶ Git Annex bringt git bei, von bestimmten Dateien nicht den *Inhalt*, sondern nur *Metadaten* zu tracken
- ▶ Git Annex merkt sich, in welchen *Repositories/PCs* welche *Inhalte* gespeichert sind
- ▶ Git Annex benutzt git zum Synchronisieren der Metadaten, div. Protokolle für die Inhalte

- ▶ Git Annex führt einen git-annex-Branch im Hintergrund. Dort ist gespeichert:
  - ▶ Repository-weite **Einstellungen** (git annex config)
  - ▶ **Beschreibungen** jedes bekannten Repositories (Festplatte1, Laptop2, Server3, etc.) (git annex info)
  - ▶ Welche Dateien will jedes Repository **bevorzugt** haben? (git annex wanted)
  - ▶ **Logbuch**: In welchem Repository ( $\hat{=}$ Datenträger) liegt welche Datei? (git annex log|whereis|list|etc.)
  - ▶ Beliebige **Metadaten** (Notizen, etc.) zu Dateien (git annex metadata)
- ▶ Der git-annex-Branch ist so gestaltet, dass er ohne Konflikte gemergt werden kann

- ▶ Jedem git-Repository kann ein Annex hinzugefügt werden:

```
yann in yann-desktop in ~/code
> git init myrepo
Leeres Git-Repository in /home/yann/code/myrepo/.git/ initialisiert
yann in yann-desktop in ~/code
> cd myrepo/
yann in yann-desktop in ../myrepo on ↳ main
> git annex init
init ok
(recording state in git...)
```



- ▶ Dateien bearbeiten/hinzufügen und mit `git annex assist` committen (und synchronisieren)
- ▶ `git annex assist` macht `git annex add`, `git add`, `git commit`, `git pull`, `git push` und kopiert Dateien herum, wenn nötig

```
yann in yann-desktop in ~/myrepo on ⌋ main
) echo markdownfile > bla.md
yann in yann-desktop in ~/myrepo on ⌋ main [?]
) echo textfile > bla.txt
yann in yann-desktop in ~/myrepo on ⌋ main [?]
) git annex assist
add bla.txt ok
add bla.md ok
(recording state in git...)
commit (recording state in git...)

[main (Root-Commit) 0724973] git-annex in yann@yann-desktop:~/code/myrepo
2 files changed, 2 insertions(+)
create mode 120000 bla.md
create mode 120000 bla.txt
ok
```

- ▶ Git Annex hat aus den Dateien **symbolische Links** gemacht
- ▶ Die Links zeigen in den versteckten Ordner unter `.git/annex/objects`

```
yann in yann-desktop in ~/myrepo on main
) ls -l
lrwxrwxrwx 186 yann yann 29 Jun 16:51 bla.md -> .git/annex/objects/GK/z0/SHA256E-s13--a0183c09845d5b4e92d4b84f89d0be281aefcd6ad654bf0f508517f15eba6cd7.md/SHA256E-s13--a0183c09845d5b4e92d4b84f89d0be281aefcd6ad654bf0f508517f15eba6cd7.md
lrwxrwxrwx 186 yann yann 29 Jun 16:51 bla.txt -> .git/annex/objects/Gp/MF/SHA256E-s9--fadb343834b0db84dbee6ca3cadbe8a6fe367caccca8958b8bc96d8379ed43af.txt/SHA256E-s9--fadb343834b0db84dbee6ca3cadbe8a6fe367caccca8958b8bc96d8379ed43af.txt
```

- ▶ Dort werden Dateien so abgelegt, dass sie **nicht gelöscht oder bearbeitet werden können**

```
yann in yann-desktop in ~/myrepo on main
) tree -l .git/annex/objects/
drwxr-xr-x - yann yann 29 Jun 16:51 .git/annex/objects
drwxr-xr-x - yann yann 29 Jun 16:51 GK
drwxr-xr-x - yann yann 29 Jun 16:51 z0
dr-xr-xr-x - yann yann 29 Jun 16:51 SHA256E-s13--a0183c09845d5b4e92d4b84f89d0be281aefcd6ad654bf0f508517f15eba6cd7.md
.r--r--r-- 5 yann yann 29 Jun 16:51 SHA256E-s13--a0183c09845d5b4e92d4b84f89d0be281aefcd6ad654bf0f508517f15eba6cd7.md
drwxr-xr-x - yann yann 29 Jun 16:51 Gp
drwxr-xr-x - yann yann 29 Jun 16:51 MF
dr-xr-xr-x - yann yann 29 Jun 16:51 SHA256E-s9--fadb343834b0db84dbee6ca3cadbe8a6fe367caccca8958b8bc96d8379ed43af.txt
.r--r--r-- 9 yann yann 29 Jun 16:51 SHA256E-s9--fadb343834b0db84dbee6ca3cadbe8a6fe367caccca8958b8bc96d8379ed43af.txt
```

**nur Leserechte!**

- ▶ Dateien können mit `git annex unlock` entsperrt und zu normalen Dateien werden




```
yann in yann-desktop in ../myrepo on ↵ main
> git annex unlock
unlock bla.md ok
unlock bla.txt ok
(recording state in git...)
yann in yann-desktop in ../myrepo on ↵ main
> ls -l
.rw-r--r-- 13 yann yann 29 Jun 16:51 📁 bla.md
.rw-r--r--  9 yann yann 29 Jun 16:51 📄 bla.txt
```

- ▶ Wenn auch neue Dateien bearbeitbar bleiben sollen: `git annex config --set annex.addunlocked true`

```
yann in yann-desktop in .../myrepo on ʘ main
) git annex config --set annex.addunlocked true
annex.addunlocked true ok
(recording state in git...)
yann in yann-desktop in .../myrepo on ʘ main
) echo image > bla.jpg
yann in yann-desktop in .../myrepo on ʘ main [?]
) git annex assist
add bla.jpg ok
(recording state in git...)
commit (recording state in git...)


[main 5f82b70] git-annex in yann@yann-desktop:~/code/myrepo
1 file changed, 1 insertion(+)
create mode 100644 bla.jpg
ok
yann in yann-desktop in .../myrepo on ʘ main
) ls
📁 bla.jpg 📄 bla.md 📄 bla.txt
```

## Locked (Symlink)

-  nicht bearbeitbar
-  Manche Programme mögen keine Symlinks
-  identische Dateien mehrmals kopiert im Repository benötigen nur **einmal** Speicherplatz

- ▶ Man kann auch automatische Regeln definieren
- ▶ z.B. `.ods`-Dateien immer unlocked:
- ▶ `git annex config --set annex.addunlocked 'include=*.ods'`

## Unlocked (normale Datei)

-  bearbeitbar
-  funktioniert mit allen Programmen
-  jede Kopie im Repository benötigt Speicher plus die Ur-Datei in `.git/annex/objects`

- ▶ Um Dateien und Metadaten zu synchronisieren, müssen *remotes* hinzugefügt werden
  - ▶ **Normale git remotes** wie z.B. GitLab, GitHub (die aber nur für Metadaten) oder eigene git-Server mit Git Annex installiert
  - ▶ **Spezielle Remotes:** ohne Metadaten, nur Datei-Inhalte, diverse Möglichkeiten
    - ▶ adb, Amazon Glacier, bittorrent, bup, ddar, directory, gcrypt, git-lfs, hook, rsync, S3, tahoe, tor, web, webdav, git, httpalso, borg, xmpp
    - ▶ GPG-Verschlüsselung und Chunking zur Dateigrößenverschleierung möglich
    - ▶ offenes Protokoll für eigene Remote-Typen

```
yann in yann-desktop in .../myrepo on main
> git annex info
trusted repositories: 0
semitrusted repositories: 3
    00000000-0000-0000-0000-000000000001 -- web
    00000000-0000-0000-0000-000000000002 -- bittorrent
    caf3b12a-f256-45e5-addb-e9251034e284 -- yann@yann-desktop:~/code/myrepo [here]
untrusted repositories: 0
transfers in progress: none
available local disk space: 9.86 gigabytes (+100 megabytes reserved)
local annex keys: 3
local annex size: 28 bytes
annexed files in working tree: 3
size of annexed files in working tree: 28 bytes
bloom filter size: 32 mebibytes (0% full)
backend usage:
    SHA256E: 3
```

```
yann in yann-desktop in ../myrepo on main
) git annex describe here "My Desktop (yann@yann-desktop:~/code/myrepo)"
describe here ok
(recording state in git...)
yann in yann-desktop in ../myrepo on main
) git annex info
trusted repositories: 0
semitrusted repositories: 3
    00000000-0000-0000-0000-000000000001 -- web
    00000000-0000-0000-0000-000000000002 -- bittorrent
    caf3b12a-f256-45e5-addb-e9251034e284 -- My Desktop (yann@yann-desktop:~/code/myrepo) [here]
```



```
yann in yann-desktp in .../myrepo on ʘ main
) git init /mnt/usb4TB/myrepo
Leeres Git-Repository in /mnt/usb4TB/myrepo/.git/ initialisiert
yann in yann-desktp in .../myrepo on ʘ main
) git remote add usb4TB /mnt/usb4TB/myrepo
yann in yann-desktp in .../myrepo on ʘ main
) git annex assist
commit
Auf Branch main
nichts zu committen, Arbeitsverzeichnis unverändert
ok
pull usb4TB
ok
push usb4TB
Objekte aufzählen: 31, fertig.
Zähle Objekte: 100% (31/31), fertig.
Delta-Kompression verwendet bis zu 4 Threads.
Komprimiere Objekte: 100% (25/25), fertig.
Schreibe Objekte: 100% (31/31), 5.15 KiB | 1.29 MiB/s, fertig.
Gesamt 31 (Delta 4), Wiederverwendet 0 (Delta 0), Pack wiederverwendet 0
To /mnt/usb4TB/myrepo
* [new branch]      main -> synced/main
* [new branch]      git-annex -> synced/git-annex
ok
```

- Der Datenträger ist jetzt im git-annex-Branch verankert

```
yann in yann-desktop in .../myrepo on └─ main
) git annex info
trusted repositories: 0
semitrusted repositories: 4
00000000-0000-0000-0000-000000000001 -- web
00000000-0000-0000-0000-000000000002 -- bittorrent
be1a2ac4-0942-464b-b54e-2d5ca44e1325 -- usb4TB
caf3b12a-f256-45e5-addb-e9251034e284 -- My Desktop (yann@yann-desktop:~/code/myrepo) [here]
```

- ▶ Auf der Festplatte ist noch nichts

```
yann in yann-desktop in ../myrepo on ↵ main
> git annex list
here
|usb4TB
||web
|||bittorrent
||||
X___ bla.jpg
X___ bla.md
X___ bla.txt
```

- ▶ Mit `git annex assist` werden die Dateien auf die Festplatte kopiert

```
yann in yann-desktop in .../myrepo on ʘ main
) git annex assist
commit
Auf Branch main
nichts zu committen, Arbeitsverzeichnis unverändert
ok
pull usb4TB
remote: Objekte aufzählen: 5, fertig.
remote: Zähle Objekte: 100% (5/5), fertig.
remote: Komprimiere Objekte: 100% (3/3), fertig.
remote: Gesamt 3 (Delta 1), Wiederverwendet 0 (Delta 0),
Entpacke Objekte: 100% (3/3), 407 Bytes | 407.00 KiB/s,
Von /mnt/usb4TB/myrepo
3ec1875..00a5416 git-annex -> usb4TB/git-annex
ok
(merging usb4TB/git-annex into git-annex...)
copy bla.jpg (to usb4TB...)
ok
copy bla.md (to usb4TB...)
ok
copy bla.txt (to usb4TB...)
ok
pull usb4TB
```

- ▶ Jetzt sind die Dateien auf der Festplatte

```
yann in yann-desktop in ../myrepo on ↵ main
> git annex list
here
|usb4TB
||web
|||bittorrent
||||
XX__ bla.jpg
XX__ bla.md
XX__ bla.txt
```

- ▶ Statt Festplatte einen Server hinzufügen (`git remote add myserver ssh://...`)
- ▶ Alle Dateien von der Festplatte löschen (`git annex drop --from=usb4TB`)
- ▶ Alle Dateien auf die Festplatte verschieben (`git annex move --to=usb4TB`)
- ▶ Erfordern, dass es mindestens 2 Kopien gibt (dann weigert sich Git Annex bei den beiden Befehlen oben) (`git annex numcopies 2`)
- ▶ Dateien Metadaten zuordnen (`git annex metadata --set notiz='wichtig!' bla.md`)
- ▶ Metadaten abfragen (`git annex metadata bla.md`)
- ▶ Ordnerstruktur entsprechend Metadaten erstellen, z.B. in Musiksammlung erste Ebene der Interpret, dann das Album (`git annex view Interpret='*' Album='*'`)
- ▶ Festlegen, dass die Festplatte nur .md-Dateien will – alles andere wird dann von dort entfernt (`git annex wanted usb4TB 'include=*.md'`)
- ▶ ...

- ▶ Manche Dateien von git tracken lassen, andere mit Git Annex

```
1 .gitattributes +
1 * annex.largefiles=((mimeencoding=binary)and(largerthan=0))
1 *.py annex.largefiles=false
2 *.sh annex.largefiles=false
3 *.ods annex.largefiles=true
~
```

- ▶ Annex-Dateien, die nicht im aktuellen git-tree mehr vorkommen finden und entfernen (oder woanders hin verschieben)

```
yann in yann-desktop in .../myrepo on ʘ main took 2s630ms
ʘ ls
  bla.jpg  bla.md  bla.txt
yann in yann-desktop in .../myrepo on ʘ main
ʘ rm bla.md
yann in yann-desktop in .../myrepo on ʘ main [x]
ʘ git annex assist >/dev/null 2>/dev/null
yann in yann-desktop in .../myrepo on ʘ main took 2s214ms
ʘ git annex unused
unused . (checking for unused data...) (checking main...)
Some annexed data is no longer used by any files:
  NUMBER  KEY
  1        SHA256E-s13--a0183c09845d5b4e92d4b84f89d0be281aefcd6ad654bf0f508517f15eba6cd7.md
(To see where this data was previously used, run: git annex whereused --historical --unused

To remove unwanted data: git-annex dropunused NUMBER

ok
yann in yann-desktop in .../myrepo on ʘ main
ʘ git annex drop --unused
drop SHA256E-s13--a0183c09845d5b4e92d4b84f89d0be281aefcd6ad654bf0f508517f15eba6cd7.md ok
(recording state in git...)
```



## ► Webinterface für Hintergrundsynchroisation à la Syncthing/Nextcloud/Dropbox

git-annex | Dashboard | Configuration | About | Files | Repository: ~/code/3d/3dprints

### Repositories

<input type="checkbox"/> yann-desktop-uni [here]	syncing enabled	actions ▾
<input type="checkbox"/> [silya]	syncing enabled	actions ▾
<input type="checkbox"/> gitlab	syncing enabled (metadata only)	actions ▾
<input type="checkbox"/> [hetzner]	syncing enabled	actions ▾

Sync your files with another device, or share with a friend.

### Transfers

*(No file transfers running)*

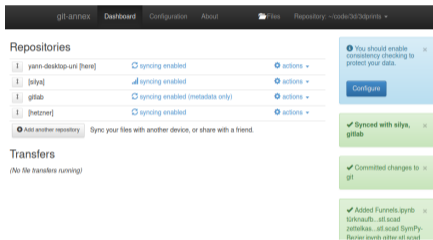
You should enable consistency checking to protect your data. ×

Synced with silya, gitlab ×

Committed changes to git ×

Added Funnels.ipynb türknauf...stl.scad zettelkas...stl.scad SymPy-Bezier.iovnb gitter.stl.scad ×

- ▶ Läuft im Hintergrund, Autostart möglich
- ▶ Im Grunde eine Endlosschleife `git annex assist` (nur effizienter)
- ▶ Fügt alle Dateien `unlocked` hinzu
- ▶ Bei `git`-getrackten Merge-Conflicts ungünstig, also am besten nur bei reinen Annex-Repos
- ▶ Kleine, einfache GUI zum Verwalten der **Remotes**, keine Dateiverwaltung



- ▶ Git Annex Homepage: <https://git-annex.branchable.com>
- ▶ DataLad – Forschungsdatenmanagement mit Git Annex: <https://datalad.org>
- ▶ DataLad/Git Annex-Konferenz nächstes Jahr wohl irgendwo in Deutschland
- ▶ Mein Thunar-Plugin für Git Annex: <https://pypi.org/project/thunar-plugins>

**Yann Büchau**

- ✉ GUZ, Uni Tübingen, Schnarrenbergstr. 94-96, 72076 Tübingen
- ☎ +49 7071 29 74777
- @ nobodyinperson@posteo.de
- 🌐 nobodyinperson.de
- 📧 fosstodon.org/@nobodyinperson
- 🐙 @nobodyinperson

