Bausteine Forschungsdatenmanagement
Empfehlungen und Erfahrungsberichte für die Praxis von Forschungsdatenmanagerinnen und -managern

A push for better RDM:

Erfahrungsbericht aus dem Einsatz von git für Forschungsdaten

Magdalene Cyra[i] Marius Politze[ii] Henning Timm[iii]

2022

Zitiervorschlag

Cyra, Magdalene, Marius Politze und Henning Timm. 2022. A push for better RDM: Erfahrungsbericht aus dem Einsatz von git für Forschungsdaten. *Bausteine Forschungsdatenmanagement. Empfehlungen und Erfahrungsberichte für die Praxis von Forschungsdatenmanagerinnen und -managern* Nr. 2/2022: S. 1-17. DOI: 10.17192/bfdm.2022.2.8435.

[i]Landesinitiative für Forschungsdatenmanagement fdm.nrw, Universität Duisburg-Essen. ORCID: 0000-0001-7738-2703
[ii]IT Center RWTH Aachen. ORCID: 0000-0003-3175-0659
[iii]Universität Duisburg-Essen. ORCID: 0000-0002-5345-7122

Abstract

Die Versionskontrollsoftware git und die Serversoftware GitLab wurden für die Softwareentwicklung konzipiert, ermöglichen aber auch kooperatives Arbeiten an Forschungsdaten entlang der FAIR-Prinzipien, eine elementare Herausforderung im Forschungsdatenmanagement (FDM). Koordiniert durch die Landesinitiative fdm.nrw wurden in den vergangenen Jahren daher Abläufe und Schulungen zum FDM mit git und GitLab erprobt und durchgeführt.
Eine Hürde bei der Verwendung von git und GitLab im FDM kann die Anwendung der zugrundeliegenden Software mittels Kommandozeilenbefehlen darstellen. Zwar wird dadurch das Verständnis der Versionierungsvorgänge erhöht, einige Forschende bevorzugen dennoch die intuitiven grafischen Interfaces, für die jedoch ebenso eine Einarbeitungszeit einzuplanen ist.
In der datengetriebenen Forschung werden die Versionsverwaltung und andere Digitalkompetenzen mittelfristig einen ebenso hohen Stellenwert wie das „wissenschaftliche Schreiben“ im Bereich der Sprachkompetenzen einnehmen müssen. Ergänzend zu bereits etablierten Schulungsformaten wurden kurze, konkrete Beispiele in einer nachnutzbaren Best-Practice-Sammlung zusammengetragen: öffentliche GitLab-Projekte realisieren und dokumentieren einzelne Anwendungsfälle mithilfe der gebotenen Werkzeuge. Hierdurch soll die Anwendungs- und Digitalkompetenz der Forschenden, aber auch der Infrastrukturmitarbeitenden und der Schulenden gesteigert werden und so zu einer besseren Umsetzung des FDM durch git und GitLab beitragen.

1 *git status* – Werkzeuge für das Versionieren von Forschungsdaten: git und GitLab

Der nachhaltige und nachvollziehbare Umgang mit jeder Art von Forschungsdaten bedarf unterstützender Werkzeuge, die es den Forschenden erleichtern Forschungsdatenmanagement (FDM) umzusetzen. Neben Werkzeugen, die disziplinspezifische Arbeitsabläufe unterstützen, gibt es andere, die besonders für bestimmte Forschungsdatentypen optimiert sind. Für textbasierte Forschungsdaten ist ein solches Werkzeug git. Die Nutzung der Plattform GitLab ermöglicht zudem das kooperative Arbeiten an textbasierten Forschungsdaten sowie die Auffindbarkeit und Nachvollziehbarkeit der Forschungsdaten. Durch öffentliche Projekte können Daten für Dritte zugänglich gemacht werden, sodass der Einsatz von git und GitLab bei der Umsetzung der FAIR-Prinzipien unterstützt.

Die Einsatzbereiche für GitLab im Hochschulbetrieb beschränken sich dabei nicht nur auf die Unterstützung im FDM, sondern sind vielfältig: Vom Einsatz in der Softwareentwicklung oder zum persönlichen Datenmanagement z. B. im Rahmen von Abschlussarbeiten, bis hin zur Organisation, Durchführung und Kollaboration in Lehrveranstaltungen profitieren Studierende und Lehrende gleichermaßen. Neben dem Einsatz im

FDM, kann es auch zum Forschungsprojektmanagement genutzt werden. Ebenso ist der Einsatz in Infrastruktureinrichtungen zur Verwaltung und Dokumentation von Prozessen möglich.

Im Folgenden liegt der Fokus auf der Nutzung von GitLab im FDM und die damit verbundenen Erfahrungen aus Beratungen und Schulungen zum Einsatz im Forschungsalltag. Schlussendlich wird eine Sammlung von Best-Practice Anwendungsfällen zur Nachnutzung vorgestellt. Mit diesem Beitrag möchten wir das grundlegende Verständnis für git und die Möglichkeiten der Nutzung im FDM darstellen und den Einstieg mithilfe der Best-Practice-Sammlung erleichtern.

2 *git checkout main* – Grundkonzepte der Versionsverwaltung

Die Software git wurde für die Versionsverwaltung von Softwarequellcodes entwickelt. Seit der Veröffentlichung 2005 hat sich git in den späten 2000er Jahren als *de facto* Standard für Versionsverwaltung im Open Source Bereich etabliert. Dabei definiert git lediglich die Techniken zur Verwaltung und Austausch von Versionsständen, sodass in den folgenden Jahren Produkte entstanden sind, die auf git aufbauen und Kollaboration unterstützen. Aufgrund der Verbreitung besonders erwähnenswert sind die Angebote von GitHub und GitLab, die das Versionskontrollsystem in eine webbasierte Plattform mit Funktionen zum Projekt-, Aufgaben- und Berechtigungsmanagement einbinden. Das Angebot GitHub der Firma GitHub Inc., einer Tochtergesellschaft von Microsoft, setzt dabei primär auf ein *Software as a Service* (SaaS) basiertes Produkt, das nach einem Freemium-Modell angeboten wird. Im Gegensatz dazu stellt das Angebot GitLab der Firma GitLab Inc. sowohl ein SaaS Angebot als auch die Software GitLab als freie Open Source Software zur Installation auf eigenen Servern zur Verfügung. Im Hochschulumfeld ist GitLab daher seit einigen Jahren weit verbreitet und wird neben der Verwaltung von Softwareprojekten vermehrt in der Hochschullehre[1], für die Verwaltung von textbasierten Forschungsdaten und von Forschungsprojekten eingesetzt.

Die Verbreitung von GitLab im Hochschulkontext wird vom ZKI Arbeitskreis „Strategie und Organisation“ in einer Umfrage betätigt[2]: GitLab belegt über alle Hochschultypen hinweg den ersten Platz in der Kategorie Projektmanagement Tools, obgleich die Projektmanagement-Funktionen eher sekundär sind. Verbreitung und Zufriedenheit mit dem Funktionsumfang werden als sehr hoch angegeben. Auch eine von IDM.NRW

[1] Küppers, Bastian, Marius Politze und Ulrik Schroeder. 2017. „Reliable e-Assessment with GIT – Practical Considerations and Implementation.“ In European Journal of Higher Education IT 2017-1, hrsg. von Johan Bergström. Umeå, Sweden. https://doi.org/10.17879/21299722960.

[2] Dreyer, Malte. 2021. „Ergebnisse der ZKI Top Trends-Umfrage des ZKI-Arbeitskreises Strategie und Organisation für das Jahr 2021.“ (Version 1). Zenodo. https://doi.org/10.5281/zenodo.4542776.

durchgeführte Umfrage[3] zeigt, dass GitLab weit verbreitet und vor allem zur hochschulübergreifenden Kollaboration genutzt wird (vgl. Abbildung 1). Trotz der weiten Verbreitung und Nachfrage nach dem Dienst stellt die Landesinitiative fdm.nrw in einer bundesweiten Umfrage im Rahmen der „AG GitLab" jedoch fest, dass kaum flächendeckende GitLab Angebote etabliert sind (vgl. Abbildung 2)[4]. Dies ist besonders brisant, da die Einrichtung kleiner, lokaler Instanzen i. d. R. niedrigschwellig durchführbar ist, dadurch aber häufig unzugängliche Datensilos entstehen. Unter anderem im Sinne des guten FDM wären daher einige wenige zentrale Instanzen, die besser in übergeordnete Infrastrukturen eingebunden sind, wünschenswert.

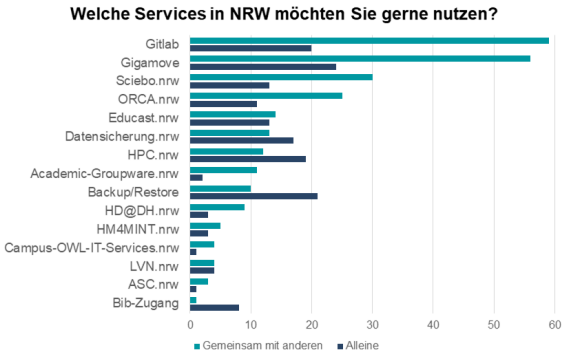

Abbildung 1: Umfrage zu NRW-weiten Services (IDM.nrw, 06/2021, N=349).

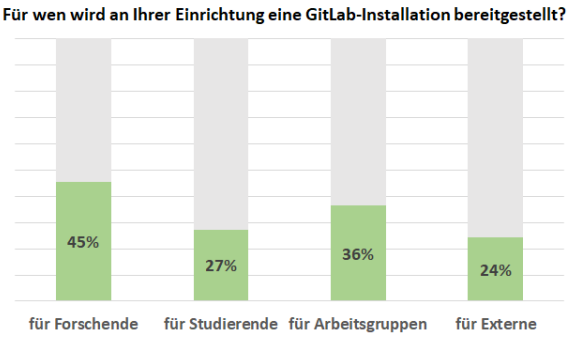

Abbildung 2: Verfügbarkeit von GitLab an Hochschulen (fdm.nrw „AG GitLab", 08/2020, N=33).

In aktuellen Kooperationsprojekten wird die Verbreitung in der Anwendung erfolgreich gemeinsam genutzter GitLab-Instanzen deutlich: GitLab ist z. B. in Abläufe der Projekte HPC.nrw[5], NFDI4Ing[6] und NHR4CES integriert und wird über die DFN-AAI hochschulübergreifend eingesetzt.

Technisch betrachtet verfolgt die Versionsverwaltung git einen dezentralen und asynchronen Ansatz (vgl. Abbildung 3): Im Dateisystem wird ein sog. *Repository* erzeugt (der dazugehörige Befehl im Kommandozeilenwerkzeug git lautet *init*), in dem die Änderungshistorie der unter Versionskontrolle befindlichen Daten verwaltet wird. Innerhalb des *Repositories* werden, transparent für die Nutzenden, die Dateiversionen vom Dateisystem als eine Kette von Änderungen, sog. *Commits* abgespeichert (*add* und *commit*). Ohne die Kette von Änderungen zu zerbrechen, kann eine neue Dateiversion immer nur an das Ende der Kette angefügt werden, sodass die gesamte Kette jederzeit nachvollzogen und intermediäre Versionen wiederhergestellt werden können.

[3] https://idm.dh.nrw/fileadmin/user_upload/idm/Bericht_Kurzumfrage.pdf, letzter Zugriff: 04.07.2022.
[4] https://www.fdm.nrw/wp-content/uploads/2020/10/Umfrage-zur-Nutzung-von-GitLab_09-2020.pdf, letzter Zugriff: 04.07.2022.
[5] https://hpc.dh.nrw/fileadmin/user_upload/hpc/Berichte/HPC_NRW_Bericht_AP2.pdf, Seite 3. Letzter Zugriff: 02.08.2022.
[6] https://nfdi4ing.de/base-services/s-2/, letzter Zugriff: 02.08.2022.

Über einen Synchronisationsmechanismus (*clone*, *push*, *fetch* und *pull*) lassen sich die gesamten Ketten von Versionshistorien zwischen dem lokalen Dateisystem und einer „entfernten" Kopie (*remote*) z. B. auf einem Server austauschen und sichern oder mit Dritten teilen. Aufgrund der Asynchronität und parallelen Bearbeitungen können beim Austausch der Ketten Konflikte auftreten, quasi parallele Stränge in den Ketten. Werden diese wieder miteinander verbunden (*merge*) reihen sie sich in die Gesamthistorie ein.

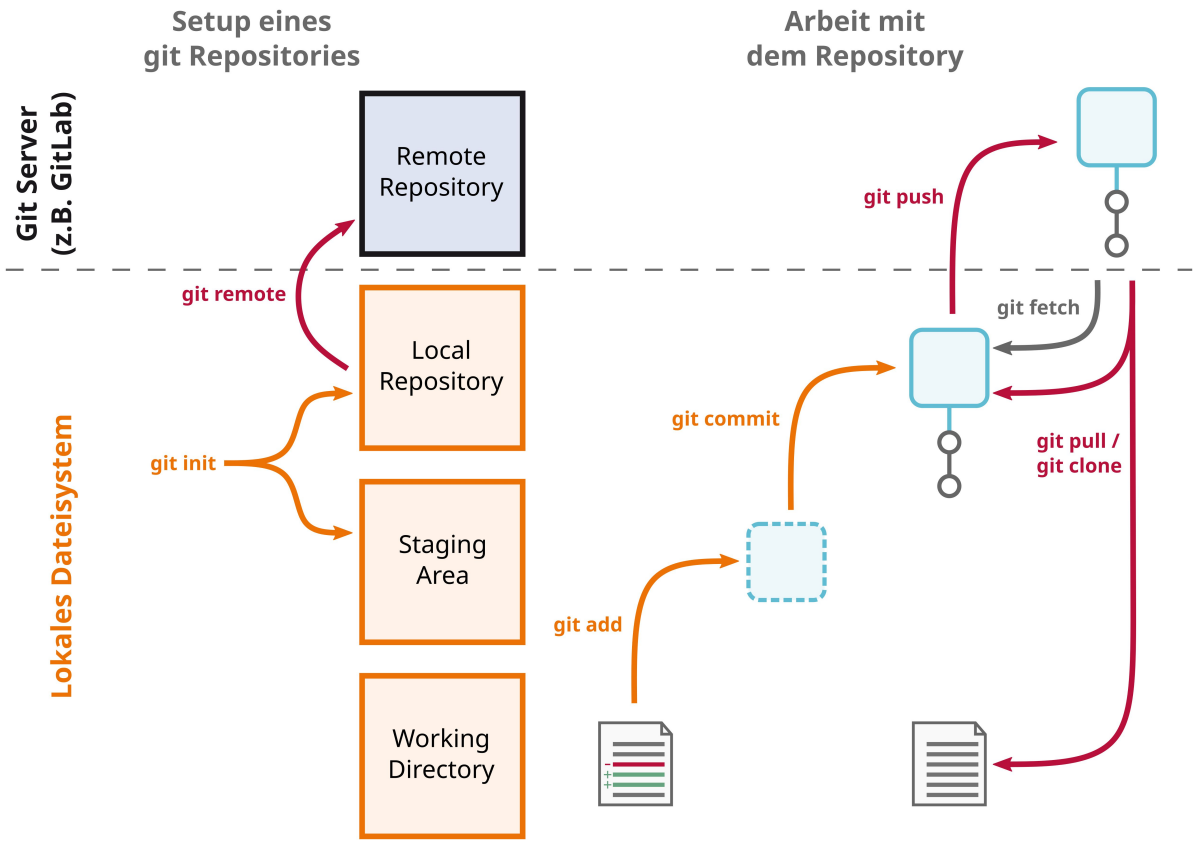

Abbildung 3: Kernablauf der Versionskontrolle mit git und einem git Server.

Basierend darauf gibt es eine Vielzahl von beliebig miteinander kompatiblen Implementierungen, die die Arbeit mit git-*Repositories* vereinfachen und um grafische Elemente erweitern, die sich auf der Kommandozeile sonst nicht realisieren lassen. Insgesamt lassen sich diese grob in vier Kategorien einteilen: Eigenständige git Clients, Dateimanager-Integrationen, Integrationen in Programmierumgebungen, Automatische Synchronisation. Tabelle 1 zeigt einige Beispiele und wesentliche Funktionsmerkmale.

Tabelle 1: Beispiele für verschiedene git Implementierungen.

Client	Funktionsmerkmale
Eigenständige git Clients	
GitHub Desktop[7] GitKraken[8]	• stellen ein eigenes Programm auf dem Computer, das genutzt werden kann, um mit der Versionsverwaltung zu interagieren.
Dateimanager-Integrationen	
TortoiseGit[9]	• stellen die Funktionen der Versionsverwaltung direkt im Dateimanager des Betriebssystems dar. Datei- und Ordnersymbole werden durch Statusanzeigen ergänzt (bspw. Ausrufezeichen bei Änderungen). • Interaktion mit der Versionsverwaltung erfolgt über das Kontextmenü. Da der Dateimanager oft das primäre Werkzeug zum Arbeiten mit Dateien auf dem Computer ist, werden diese meist intuitiv genutzt.
Integrationen in Programmierumgebungen	
VSCode[10] RStudio[11] Jupyter[12]	• kombinieren die Versionsverwaltung mit Werkzeugen zum Editieren der Dateien. • Änderungen und Versionen können nicht nur auf Dateiebene, sondern direkt in der Editoransicht angezeigt werden, was eine verbesserte Integration der Versionsverwaltung darstellt.
Automatische Synchronisation	
SparkleShare[13] GitDoc[14]	• verbirgt die Versionierung nahezu vollständig vor den Nutzenden. Bei jeder Änderung auf dem Dateisystem wird eine neue Version erstellt. • Nutzende haben deutlich weniger Kontrolle, i. d. R. können keine Versionskommentare eingegeben werden. • Zudem kann die Hemmschwelle für die Nutzenden steigen, komplexe Szenarien wie den Zugriff auf frühere Versionen oder parallele Bearbeitungsstränge (sog. branches) umzusetzen.

[7] https://desktop.github.com/, letzter Zugriff: 04.07.2022.
[8] https://www.gitkraken.com/, letzter Zugriff: 04.07.2022.
[9] https://tortoisegit.org/, letzter Zugriff: 04.07.2022.
[10] https://code.visualstudio.com/, letzter Zugriff: 04.07.2022.
[11] https://support.rstudio.com/hc/en-us/articles/200532077-Version-Control-with-Git-and-SVN, letzter Zugriff: 04.07.2022.
[12] https://github.com/jupyterlab/jupyterlab-git, letzter Zugriff: 04.07.2022.
[13] https://www.sparkleshare.org/, letzter Zugriff: 04.07.2022.
[14] https://github.com/lostintangent/gitdoc, letzter Zugriff: 04.07.2022.

Auch mit der Unterstützung von grafischen Bedienelementen in Ergänzung zur Kommandozeile zeigt sich, dass Workflows zur Versionsverwaltung aus der Softwareentwicklung nicht immer passgenau im FDM sind, oder, dass weitere Automatisierungen möglich und nötig sind. So gibt es FDM-spezifische Überbauten wie RePlay-DH[15], GIN Tonic[16], DataLad[17] oder CONQUAIRE[18], die den Umgang mit Forschungsdaten in git *Repositories* oder in der Umgebung von GitLab verbessern.

Technisch gesehen ist die Versionsverwaltung sehr komplex, sodass zum Teil mit einer erheblichen Einarbeitungszeit zu rechnen ist. Aufbauend auf den Grundlagen sind oft weitere Kenntnisse in der Konfiguration von git und GitLab notwendig, um das volle Potenzial auszuschöpfen. Die Definitionen von fachspezifischen Arbeitsabläufen können die Automatisierung oder das Qualitätsmanagement ermöglichen und darüber hinaus zwischen Arbeitsgruppen ausgetauscht und nachgenutzt werden, wenn diese entsprechend dokumentiert sind.

3 *git rebase fdm* – Ein Werkzeug aus der Softwareentwicklung für das Forschungsdatenmanagement

Der Fokus von git auf Textdateien ist zentral verankert, z. B. bei Source Code. Änderungen werden anhand von geänderten Zeilen in Textdateien verfolgt. Dateien, die nicht im Textformat vorliegen, sog. Binärdateien – wie Bilder oder komprimierte Dateien – können von git zwar versioniert werden, jedoch ist das zeilenweise Verfolgen von Änderungen nicht möglich. Im Kontext des FDMs ist git sehr gut geeignet zur Verwaltung offener und menschenlesbarer Dateiformate. Dazu zählen beispielsweise Datenreihen in CSV-Dateien, Metadatenverwaltung zu diesen Daten in XML oder JSON-Dateien und Software zur reproduzierbaren Analyse Source Code. Auch die Features von GitLab sind auf Softwareentwicklung und -wartung ausgerichtet, können aber, genau wie bei git, gut im Kontext FDM eingesetzt zu werden. Gerade bei nichttextuellen Daten kommen git und GitLab jedoch an technische Grenzen, die über Erweiterungen des git Protokolls wie GitLSF[19] oder GitAnnex[20] bis zu einem gewissen Grad umgangen werden können, aber gegebenenfalls zusätzliche Infrastruktur benötigen.

[15] Hermann, Sibylle, Uli Hahn, Markus Gärtner und Florian Fritze. 2018. „Nachträglich ist nicht gleich nachnutzbar: Ansätze für integrierte Prozessdokumentation im Forschungsalltag." 32-45 Seiten / o-bib. Das offene Bibliotheksjournal / Herausgeber VDB, Bd. 5 Nr. 3 (2018). https://doi.org/10.5282/o-bib/2018H3S32-45.

[16] Julien Colomb. 2020. „GIN-tonic: Collaboration in research made easy. Neuromatch." Zenodo. https://doi.org/10.5281/zenodo.4159325.

[17] Halchenko, Yaroslav, Kyle Meyer, Benjamin Poldrack, Debanjum Solanky, Adina Wagner, Jason Gors, Dave MacFarlane et al. 2021. „DataLad: distributed system for joint management of code, data, and their relationship." JOSS 6 (63): 3262. https://doi.org/10.21105/joss.03262.

[18] Cimiano, Philipp, Christian Pietsch und Cord Wiljes. 2021. Studies in Analytical Reproducibility: the Conquaire Project: Universität Bielefeld. https://doi.org/10.4119/unibi/2942780.

[19] https://git-lfs.github.com/, letzter Zugriff: 04.07.2022.

[20] https://git-annex.branchable.com/, letzter Zugriff: 04.07.2022.

In einem GitLab-Projekt, dessen Kern ein git *Repository* zur Datenversionierung bildet, können über den Zugang verschiedene Nutzende und deren Zusammenarbeit organisiert werden. Ein Beispiel dafür sind *Issues*, kurze Dokumente, um Aufgaben zu beschreiben, die mittels Schlagwörtern kategorisiert und Personen zugewiesen werden können. Ist eine Aufgabe abgeschlossen, wird das zugehörige *Issue* als „geschlossen" markiert. Darüber hinaus können durch *Issues* z. B. die Verantwortlichkeit für Datensätze oder Proben modelliert werden. Zusätzlich bietet jedes Projekt ein Wiki, in dem Informationen zum Projekt strukturiert abgelegt und referenziert werden können.

Ferner bietet GitLab Strukturen, die die gemeinsame Arbeit an denselben Dateien und die Auflösung ggf. auftretender Konflikte unterstützen. Bei einem *Merge Request* (MR) werden Änderungen eines Nutzenden in das Haupt-*Repository* integriert. Dabei werden Konflikte aufgezeigt, und es können automatische Verfahren zur Qualitätssicherung erforderlich werden. Zum Beispiel kann mittels automatisierter Tests überprüft werden, ob eine hinzugefügte JSON-Datei mit Metadaten einem vorgegebenen Schema entspricht. Der *Merge* findet nur statt, wenn die JSON-Datei dem Schema genügt.

Solche automatisierten Tests werden mit sog. Pipelines – Abläufen von Befehlen, die z. B. auf virtuellen Maschinen (GitLab Runner) ausgeführt werden – realisiert. Ausgelöst werden diese durch Ereignisse wie z. B. dem Eingang eines *Commits* oder der Änderung einer bestimmten Datei im *Repository*. Pipelines können im Kontext FDM darüber hinaus noch mehr leisten: Eine weitere Funktion ist das automatische Erzeugen von Dateien (sog. Artefakte), wie z. B. Plots der erzeugten Daten, den Inhalt einer Webseite, oder einen aus gesammelten Daten aggregierten und normierten Datensatz. Weiterhin kann im Rahmen einer Pipeline direkt mit externen Servern kommuniziert werden, z. B. um ein Artefakt direkt in einem Datenrepositorium abzulegen. Ein als Artefakt erzeugtes HTML-Dokument kann über die Funktion GitLab Pages direkt als Webseite angeboten werden.

Diese fortlaufende und direkte Überprüfung und Verarbeitung der in GitLab verwalteten Daten wird unter dem Begriff kontinuierliche Integration (*Continuous Integration*, CI) zusammengefasst. Mit diesem Werkzeug modellierte Abläufe erreichen eine hohe Standardisierung, da sie für alle Projektmitarbeitenden gleich durchgeführt werden. Zudem wird eine Reproduzierbarkeit erreicht, da die im CI durchgeführten Abläufe zusammen mit den Daten versioniert werden. Die Pipelines lassen sich so prinzipiell immer wieder automatisiert und identisch ausführen, um Ergebnisse zu reproduzieren.

Durch die Verbindung der Versionsverwaltung git und den Verwaltungsfunktionen von GitLab lassen sich mit überschaubarem zusätzlichem Aufwand wissenschaftliche Abläufe entlang der FAIR-Prinzipien[21] etablieren. Tabelle 2 zeigt einige Überschneidungen zwischen git, GitLab und den FAIR-Prinzipien.

[21] Wilkinson, Mark D., Michel Dumontier, I. Jsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg et al. 2016. „The FAIR Guiding Principles for Scientific Data Management and Stewardship." Scientific data 3. https://doi.org/10.1038/sdata.2016.18.

Tabelle 2: Überschneidungen zwischen git bzw. GitLab und den FAIR-Prinzipien.

FAIR Prinzip		Übertragung nach git und GitLab	
		Überschneidung	Ausprägung
F	F1	gering	Über die eindeutige Projekt ID und den Versions-Hash können in einer GitLab-Instanz Dateiinhalte abgerufen werden, sofern Nutzende zum Zugriff berechtigt sind. GitLab bietet keine global eindeutigen Identifier.
	F2	keine	GitLab bietet wenige generelle Metadatenfelder insb. Projektbeschreibung und Schlagworte, die frei belegt werden können.
	F3	hoch	Projekte erhalten eine eindeutige Projekt ID und eine URL, mit der jedes Projekt eindeutig innerhalb einer GitLab-Instanz identifiziert werden kann. Innerhalb des Projekts und im git log können Dateien in allen Versionen über einen sog. Versions-Hash eindeutig identifiziert werden.
	F4	gering	Projekte in GitLab sind innerhalb einer Instanz in einer Suche verfügbar und können durch (Meta-) Suchmaschinen indiziert werden.
A	A1	hoch	Über die GitLab API lassen sich Daten und Metadaten direkt mit der lokalen ID abrufen.
	A1.1	hoch	git bietet standardisierte Protokolle auf Basis von HTTPS und SSH zum Abrufen der Daten.
	A1.2	hoch	Daten können sowohl öffentlich als auch über Authentifizierungsmechanismen geschützt zugänglich gemacht werden.
	A2	keine	GitLab-Projekte können jederzeit gelöscht werden und sind dann nicht mehr auffindbar.
I	I1	hoch	Best-Practices von GitLab werden i. d. R. in offenen, textbasierten Datenformaten (Text, Markdown, YAML) umgesetzt und können so Plattform- und Softwareunabhängig weiterverwendet werden.
	I2	gering	Weitere textbasierte Datenformate können sehr gut mit git verwaltet werden und aus einigen Formaten (z. B. YAML-Frontmatter in Markdown) lassen sich disziplinspezifische Metadaten von GitLab automatisch extrahieren.
	I3	gering	Best-Practices für die Erfassung von Metadaten in Projekten: Spezielle Dateien (z. B. README.md, LICENSE) werden von GitLab erkannt und in strukturierte Projektmetadaten umgewandelt.
R	R1	gering	Weitere Metadaten jenseits der Softwareentwicklung können in Metadatendateien (z. B. im YAML-Format) abgelegt werden.
	R1.1	hoch	Starke Orientierung im Bereich Open Source Software: Möglichkeiten zur Lizenzierung ist fest in Repositories verankert und direkt in GitLab sichtbar.
	R1.2	hoch	Die durchgängige Versionierung erlaubt eine Nachverfolgbarkeit (Provenienz) der Daten.
	R1.3	gering	Standards folgen im Wesentlichen der Domäne „Softwareentwicklung“, sind aber teilweise übertragbar.

Aller Überschneidungen zum Trotz ist der bloße Einsatz von git kein Garant für die Erfüllung aller FAIR-Prinzipien. Zu beachten ist insbesondere, dass GitLab zu Projekten im Wesentlichen Metadaten erfasst, die für die Softwareentwicklung relevant sind: Lizenzen, Autoren, Titel, Testfallabdeckung, u. ä. Eine Erweiterung um disziplinspezifische („rich“) Metadaten (F2, I2, R1.3), sowie der Nachweis und die langfristige Speicherung im Sinne einer wissenschaftlichen Publikation z. B. in einem geeigneten Datenrepositorium (A2) sind zusätzlich erforderlich. Zu letzterem zählen insbesondere geeignete, global eindeutige persistente *Identifier* (PID) und sog. *Tombstones*, die bestehen bleiben, wenn ein Repository in einer GitLab-Instanz gelöscht werden sollte (F1, A2). Entsprechende disziplinspezifische Best-Practices z. B. zu Metadatendateien oder eine automatische Übertragung an ein Datenrepositorium wie beispielsweise Zenodo können jedoch mit CI forciert werden, wenn die Nutzenden ihr GitLab-Projekt entsprechend konfigurieren. Eine Indizierung in (Meta-)Suchmaschinen erfordert mitunter weitere Metadaten, die z. B. im YAML-Format angegeben werden können. Eine solche Anbindung und Aufnahme von git-*Repositories* in einen Suchindex wird beispielsweise durch das OERSI Projekt[22] im Kontext von *Open Educational Resources* demonstriert[23]. Einschränkend ist zu erwähnen, dass dieses Vorgehen lediglich mit GitLab-Instanzen realisierbar ist, die global zugreifbar bzw. bei der Suchmaschine registriert sind. Eine zunehmende Fragmentierung in sehr kleine, lokale GitLab-Instanzen ist insbesondere für Suchmaschinen problematisch, was aber ebenso für andere lokale Datenrepositorien gilt.

Eine Einstiegshürde, die von Forschenden bei der Verwendung von git und GitLab als FDM-Werkzeug im Rahmen von Schulungen und Beratungen wahrgenommen wurde, ist die für viele Fachbereiche ungewohnte Verwendung der zugrunde liegenden Software. Insbesondere die Versionierung mit git wird von Forschenden ohne Hintergrund in der Softwareentwicklung als schwierig wahrgenommen. Zusätzlich zu den oben beschriebenen grafischen git-Anwendungen kann an dieser Stelle jedoch GitLab als einheitliches Interface agieren, das Forschenden einen niedrigschwelligen Einstieg in die Arbeit mit git ermöglicht. GitLab kann ein einheitliches Portal für Forschende bilden und eine einheitliche Schulung ermöglichen, die nicht von z. T. betriebssystemspezifischer Software abhängt.

4 *git remote add nrw* – Erfahrungen aus der Standortübergreifenden Zusammenarbeit in NRW

Die Landesinitiative für Forschungsdatenmanagement fdm.nrw verfolgt das Ziel, Entwicklungen im Bereich FDM in NRW zu bündeln und Kooperationen zu fördern. Ne-

[22] https://oersi.org/, letzter Zugriff: 04.07.2022.
[23] https://gitlab.com/oersi/oersi-import-scripts/-/blob/master/GITLAB.md, letzter Zugriff: 04.07.2022.

ben der Aufgabe als Informationsdrehkreuz betreut die Landesinitiative fdm.nrw auch NRW-weite Arbeitsgruppen zu spezifischen Themen.

Im Rahmen des Kick-Off Workshops für die Entwicklung von hochschulübergreifenden FDM-Basisdienstleistungen in NRW wurde im Februar 2019 das Interesse bekundet, zu GitLab zusammenzuarbeiten und sich auszutauschen[24]. Seitdem kommen Vertreter:innen aus Infrastruktureinrichtungen und interessierte Forschende in der Arbeitsgruppe GitLab (AG GitLab) zusammen. In den von der Landesinitiative fdm.nrw organisierten zunächst physischen, mittlerweile virtuellen Treffen der AG GitLab wurde ein NRW-weites Netzwerk etabliert, das sich sowohl zu technischen Aspekten, wie auch zu Einsatzszenarien in der Forschung und unterstützender Infrastruktur austauscht. Die Besprechung und Umsetzung von Workflows mündeten in Best-Practices, die geteilt und diskutiert werden.

Neben den auf NRW beschränkten Treffen der AG GitLab, werden übergreifende Treffen zur Nutzung von git und GitLab im FDM durchgeführt[25]. Die virtuellen Veranstaltungen stoßen auf großes Interesse und ermöglichen es den Teilnehmenden sich über den disziplinspezifischen Einsatz von GitLab auszutauschen und sich untereinander zu vernetzen.

Die hohe Nachfrage nach Vernetzung geht damit einher, dass GitLab an immer mehr Standorten eingesetzt wird und der damit wachsende Nutzendenkreis einen steigenden Bedarf nach einführenden Schulungen hat. In der AG GitLab wurden deshalb Konzepte zur Durchführung von einführenden Workshops besprochen und geteilt. Die Landesinitiative fdm.nrw bietet gemeinsam mit NFDI4Ing und NHR4CES einen Workshop für Infrastrukturmitarbeitende und Forschende an.

5 *git help* – Schulungskonzepte für den Einstieg in git und GitLab

5.1 GitLab Workshop als Flipped Classroom

Die Veranstaltung „Einstieg ins Forschungsdatenmanagement mit git und GitLab“ setzt auf das Flipped Classroom Format mit einer Vorbereitungs- und einer Interaktionsphase: Vor der Veranstaltung werden den Teilnehmenden online-Lehrmaterialien in einem GitLab-Projekt zur Einführung in die Grundlagen bereitgestellt. Diese umfassen Lernvideos mit Lernstandkontrollen sowie Anleitungstexte für die theoretischen Grundlagen. Damit soll erreicht werden, dass sich die Teilnehmenden im Vorfeld ein Basiswissen angeeignet haben. Während der Interaktionsphase werden Konzepte und

[24] https://www.fdm.nrw/index.php/fdm-nrw/fdm-prozessbegleitung/, letzter Zugriff: 04.07.2022.
[25] https://www.fdm.nrw/index.php/fdm-mit-gitlab/ und https://www.fdm.nrw/index.php/fdm-mit-gitlab-2/, letzter Zugriff: 04.07.2022.

Best-Practices präsentiert, diskutiert und von den Teilnehmenden direkt nachvollzogen. In der zweiten Lerneinheit werden typische Anwendungsszenarien als Übungseinheit in einer Art Rollenspiel nachgestellt.

Die Rückmeldungen der Teilnehmenden lassen auf eine durchweg positive Perzeption des Flipped Classroom Formats schließen. Dem sehr technischen Anwendungsgebiet zum Trotz konnten in der Vorbereitung die wesentlichen Probleme, bspw. bei der Installation der Software, von den Teilnehmenden individuell gelöst werden. Die z. T. sehr unterschiedlichen Vorkenntnisse der Teilnehmenden, die bei dieser Art von überfachlichen Qualifikationen sehr häufig sind, können nicht immer gänzlich ausgeglichen werden – dennoch sind die Rückmeldungen insgesamt positiv. Das Format wird im Rahmen des Weiterbildungsprogramms der Landesinitiative fdm.nrw fortwährend weiterentwickelt und unter Open Access Lizenz[26] veröffentlicht.

5.2 Die git Workshops im Rahmen der Carpentries

Für die Vermittlung der Grundlagen kann auf die, als *Open Educational Resources* verfügbaren, Schulungsmaterialien und -Formate der *Software Carpentries* zurückgegriffen werden. Didaktisch folgen die Workshops im Carpentries-Format dem Prinzip des *Participatory Live Coding*. Dabei durchlaufen die Trainer ein grob abgestecktes Skript in dem die wesentlichen Abläufe und Befehle direkt, „live“, erklärt und von allen ausgeführt werden. So können die Lernenden das Erklärte zeitgleich anwenden, indem die Befehle in der eigenen Umgebung mit ausgeführt und nachvollzogen werden. Die Workshops sind ursprünglich für die Präsenzlehre entwickelt, es hat sich jedoch in der jüngsten Vergangenheit gezeigt, dass das Format in der synchronen Distanzlehre – online – erfolgreich eingesetzt werden kann.

Im Grundlagenworkshop „*Version Control with Git*“ wird der Umgang mit der Versionsverwaltungssoftware git gelehrt, um git-*Repositories* im lokalen Dateisystem mittels Kommandozeile zu nutzen. Zusätzlich werden Kollaborations- und Hostingoptionen auf git-Servern, insbesondere GitHub und die Einbindung in Open Science behandelt. Eine Vertiefung der git-Kenntnisse und anderer Digital- und Datenkompetenzen sind in den Unterbereichen „Software“, „Data“ und „Library“ möglich. Damit kann eine Grundlage für Disziplin- oder Forschungsgruppenspezifische Arbeitsabläufe geschaffen werden. Inhaltlich profitieren die Carpentry-Workshops von der kontinuierlichen Weiterentwicklung durch die weltweite Community.

[26] https://git.rwth-aachen.de/nfdi4ing/education/gitlab-workshop, letzter Zugriff: 04.07.2022.

6 *git pull examples* – Eine Sammlung von Best-Practices im Einsatz von GitLab

Im Rahmen einer Veranstaltung zusätzlich zu den Konzepten der Versionsverwaltung mit git und dem Projektmanagement mit GitLab auch direkt Anwendungsfälle im Bereich FDM zu vermitteln stellt die Teilnehmenden der Schulungen vor eine große Hürde. Um die abstrakten Konzepte, die im Rahmen der Schulung vermittelt wurden, auf den FDM-Kontext anzuwenden haben sich kurze, konkrete Beispiele als hilfreiches Mittel erwiesen. Im Falle von git und GitLab können diese durch öffentliche GitLab-Projekte realisiert werden, die bestimmte Anwendungsfälle, deren Lösung mithilfe der von der Software gebotenen Werkzeuge und eine detaillierte Dokumentation enthalten.

Um Forschenden den Einstieg in die Arbeit mit git und GitLab im FDM zu erleichtern wurde im Rahmen der AG GitLab eine Sammlung solcher Beispielprojekte angelegt[27]. Diese illustrieren jeweils einen konkreten Anwendungsfall in Form eines *Minimal Working Example* und können als Blaupause nachgenutzt werden. Die initialen Anwendungsfälle beruhen dabei u. a. auf Themen, die in Schulungen und Beratungen mit Forschenden diskutiert wurden. Ein Index-Projekt[28] dient als Übersicht der gesammelten Beispiele und verlinkt weitere externe Ressourcen. Insbesondere im Bereich Datenverifikation, kollaborativem Arbeiten mit verteilten Datensätzen und der niedrigschwelligen Erstellung von Webseiten (z. B. zur Darstellung von Datensätzen oder anderen Ergebnissen) können die von GitLab gebotenen Werkzeuge von Forschenden eingesetzt werden. Für die Nachnutzung können die Beispielprojekte entweder als Vorlage für die Konfiguration eines Projekts genutzt werden oder ein geklontes *Repository* durch das Ändern der *remote* in ein eigenes Projekt eingebunden werden. Zwei Projekte, die das Spektrum der Anwendungsfälle illustrieren, sind die JSON-Verifikation[29] und die Erstellung und Visualisierung eines kollaborativen FAIR-Datensatzes[30].

Bei der JSON-Validierung wird ein zentrales Problem des FDM adressiert: die Validierung einer dem git *Repository* hinzugefügten JSON-Datei gegen ein vorliegendes JSON-Schema. Dies tritt, unter anderem, beim Abgleich von Eingaben mit einem Metadatenschema auf (z. B. um zu prüfen, ob eine angegebene ORCID iD dem erwarteten Aufbau entspricht). Realisiert wurde dieser Anwendungsfall mithilfe einer CI-Pipeline, die nach jedem *Commit* im git *Repository* einen automatisierten Test ausführt. Diese prüft mithilfe der Software jsonschema[31] alle vorliegenden JSON-Dateien und gibt über GitLab[32] Feedback an die Nutzenden (vgl. Abbildung 4). Wird das Projekt so kon-

[27] https://gitlab.com/fdm-nrw-gitlabexamples, letzter Zugriff: 04.07.2022.
[28] https://gitlab.com/fdm-nrw-gitlabexamples/index, letzter Zugriff: 04.07.2022.
[29] https://gitlab.com/fdm-nrw-gitlabexamples/jsonschemavalidation, letzter Zugriff: 04.07.2022.
[30] https://gitlab.com/fdm-nrw-gitlabexamples/pythondatavisualization, letzter Zugriff: 04.07.2022.
[31] https://github.com/Julian/jsonschema, letzter Zugriff: 04.07.2022.
[32] https://gitlab.com/fdm-nrw-gitlabexamples/jsonschemavalidation/-/pipelines, letzter Zugriff: 04.07.2022.

figuriert, dass Beiträge zum Main-*Branch* des *Repositories* nur in Form von MRs geschehen dürfen, kann verhindert werden, dass JSON-Dateien, die nicht dem Schema entsprechen, hinzugefügt werden. Dieser Ablauf ist durch geringe Anpassung auf diverse FDM-relevante Formate (wie z. B. XML oder CSV) übertragbar, wodurch er sich gut als allgemeines Beispiel eignet.

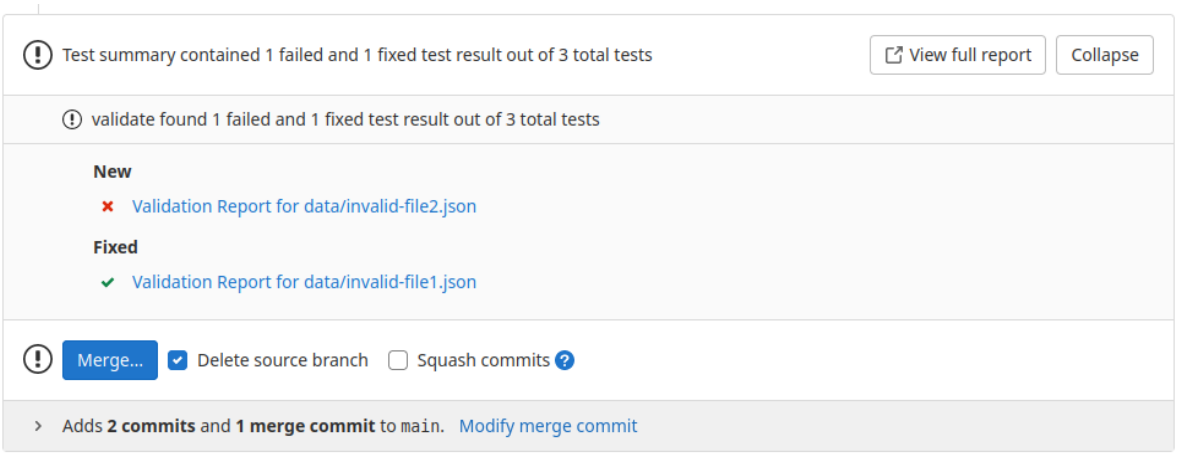

Abbildung 4: Feedback der JSON-Validierung. Durch die letzte Änderung wurden die Fehler in der Datei invalid_file1.json behoben, allerdings einen neuen Fehler in der Datei invalid_file2.json erzeugt. Ein Klick auf den Link zur fehlerhaften Datei zeigt den genauen Fehler an.

Ein weiterer FDM-spezifischer Anwendungsfall ist die kollaborative Sammlung und Verarbeitung eines FAIR-Datensatzes. In diesem Beispiel werden von Nutzenden Messpunkte für einen gemeinsam erhobenen Datensatz im CSV-Format beigesteuert. Als Beispieldaten dienen die Messungen des Gewichts von Kaffeebohnen. Die Beiträge werden in diesem Projekt über MRs realisiert, bei denen die Formatierung der Datensätze und die Formatierung des Source Codes mit automatisierten Test sichergestellt werden. Zusätzlich werden in diesem Beispiel die gesammelten Daten zusammengeführt und der aktuelle Stand des Datensatzes visualisiert. Wie bei der Validierung kommt hier CI zum Einsatz, die in diesem Fall einen Analyse-Workflow aufruft. Dieser aggregiert die gesammelten Daten und erzeugt Grafiken (z. B. über die Verteilung der Gewichte), die in Form von Artefakten zur Verfügung gestellt werden (vgl. Abbildung 5). Zusätzlich zum Beispielcharakter dieses Projekts kann der dort gepflegte Datensatz von Nutzenden erweitert werden. Die dafür notwendige Dokumentation der Daten (Codebuch, Messprotokoll) sowie der Ablauf um Messwerte beizutragen sind in der README des Projekts beschrieben. Dadurch kann dieses Projekt als lebendiges Beispiel agieren.

Im Kontext FDM eröffnen sich sehr diverse Anwendungsmöglichkeiten für git und GitLab, sodass die im Zuge dieses Projekts getroffene Auswahl an Beispielen (vgl. Tabelle 3) keineswegs als vollständig betrachtet werden. Um einen weiteren Aspekt der Nutzung von Kollaborationsplattformen wie GitLab zu illustrieren und zu nutzen, bietet

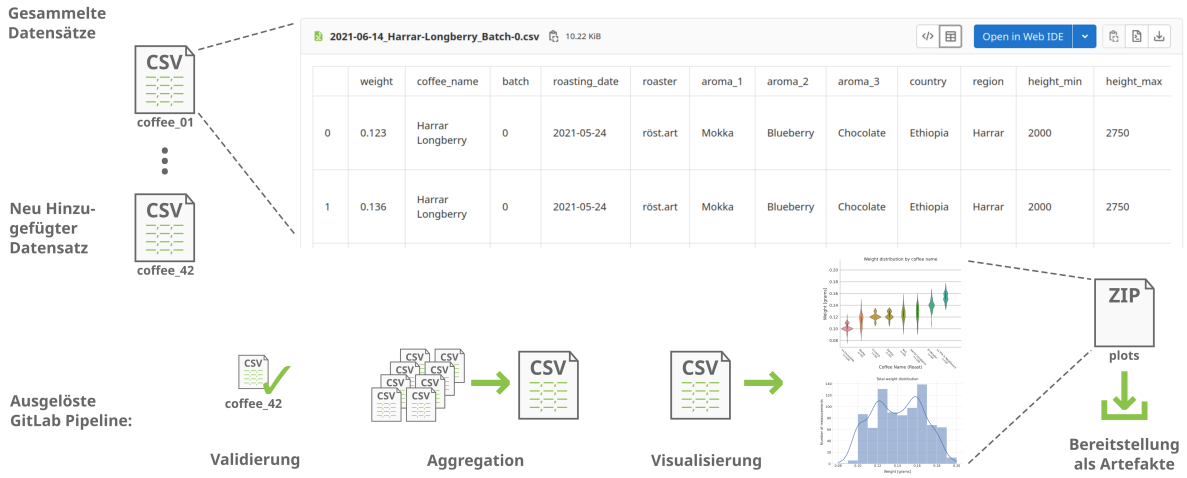

Abbildung 5: Illustration der verwendeten CI-Pipeline. Wird mittels eines MR eine neue Datei mit Messwerten hinzugefügt muss diese zuerst einer Validierung genügen. Soll dieser MR gemerged werden, müssen im Analyse-Workflow alle Messwert-Dateien zusammengeführt und visualisiert werden. Das Ergebnis dieser Visualisierung wird als Artefakt zum Download zur Verfügung gestellt.

das Projekt den Nutzenden die Möglichkeit zur Mitarbeit. Zum einen können Weiterentwicklungen der gesammelten Beispiele direkt über die Werkzeuge von GitLab (*Issues*, MRs, . . .) organisiert werden. Außerdem kann das zentrale Projekt verwendet werden, um weitere Beispiele hinzuzufügen. Dies könnte wie folgt ablaufen: Eine Nutzerin erstellt ein *Issue* im Index-Projekt und beschreibt einen Anwendungsfall, der aufgenommen werden soll. Sie kann dann ein Beispielprojekt implementieren, das gemeinsam mit Mitgliedern des Projekts entwickelt und dokumentiert wird. Abschließend wird das Projekt in die Beispiel-Sammlung aufgenommen und die beteiligten Nutzenden als *Contributor* auf der Projektseite hinzugefügt.

Tabelle 3: Liste der aktuell in der Sammlung auf https://gitlab.com/fdm-nrw-gitlabexamples enthaltenen Beispiele. Um ein breiteres Spektrum an Anwendungsfällen abzudecken sind Nutzende angehalten, weitere Beispiele aus bisher nicht behandelten Domänen zum Projekt hinzuzufügen und existierende Beispiele, mit Hilfe von MRs, zu verbessern.

Projektname	**Behandelte Aspekte**
BackslidePresentations	Erstellen von Folienpräsentationen. Anzeige der Folien als GitLab Pages.
DocsifyDocumentation SphinxDocumentation	Erstellen einer Dokumentations-Website mit Hilfe von Docsify oder Sphinx und deren Hosting als GitLab Pages.

FAIR Data Collection Analysis and Visualization	Kollaboratives Sammeln eines Datensatzes organisiert über MRs und Issues. Analyse und Visualisierung des gesammelten Datensatzes mittels CI-Pipelines.
IssueTemplates	Verwendung von Issue Templates zur Projektorganisation.
JSONSchemaValidation	Validierung von JSON-Metadaten mittels eines JSON-Schemas.
REUSESoftwareLicense-Validation	Prüfung, ob alle Dateien des Projekts eine freie Lizenz nach der REUSE-Spezifikation aufweisen.

7 *git push* – FAIRbessertes FDM mit git und GitLab

Für die zunehmend datengetriebene Forschung werden Digitalkompetenzen wie der strukturierte Umgang mit Daten und damit auch die Versionsverwaltung zukünftig einen immer höheren Stellenwert einnehmen. Generell gilt für den Aufbau der Daten- und Digitalkompetenzen der Forschenden, sowie für die wachsende Gruppe der Data Stewards (und ähnlicher), dass für die Adaption disziplinspezifischer Arbeitsabläufe eine Einarbeitungszeit einzuplanen ist. Das in der Softwareentwicklung weit verbreitete Werkzeug zur kollaborativen Arbeit, git, kann für viele Anwendungsfälle im FDM verwendet werden und bereits heute kann hier auf einen großen Erfahrungsschatz im Einsatzbereich FDM aufgebaut werden. Mit git und GitLab lassen sich viele FDM Prozesse sowie die Umsetzung der FAIR-Prinzipien gut abdecken, sodass Forschenden ein starkes FDM-Werkzeug zur Verfügung steht.

Eine Plattform zum Austausch von Erfahrungen zum Einsatz von git und GitLab im FDM zwischen Forschenden, Lehrenden und Infrastrukturanbietern ist wichtig und wird im Rahmen der AG GitLab der Landesinitiative fdm.nrw in verschiedenen Veranstaltungsformaten geleistet. Dadurch kann angeeignetes Wissen weitergegeben und von gegenseitiger Erfahrung profitiert werden; Betriebsparameter können an die Nutzung angepasst, Schulungen können disziplinspezifische Beispiele aufnehmen und Anwendungsfälle können um bisher unbekannte Techniken erweitert werden. Dabei wird immer wieder festgestellt, dass sich Ansätze durchaus disziplinübergreifend übertragen lassen, sodass neben dem direkten Austausch auch eine nachvollziehbare Dokumentation eben dieser Best-Practices notwendig ist.

Genau das leistet die vorgestellte Sammlung von Best-Practices: Die Nutzung und Umsetzung spezifischer Workflows ist nicht trivial, sodass diese Sammlung die Nachnutzung und den Einstieg erleichtert und aufzeigt, inwieweit GitLab im Kontext FDM genutzt werden kann. Die Sammlung lässt sich zudem gut in Schulungen einbinden und dient den Teilnehmenden auch nach der Schulung als zentrale Anlaufstelle. Durch

die Projektmanagement- und Kommunikationsfunktionen von GitLab ist die weitere Vernetzung der interessierten Anwendenden möglich. Im Sinne des Open-Source-Gedanken sind alle Best-Practices frei zugänglich und sind offen für Beiträge und Verbesserungen aus der Community.

Abschließend kann zusammengefasst werden, dass git und GitLab initial für einen anderen Kontext entwickelt wurden, sich aber sehr gut in FDM-Arbeitsabläufe integrieren lassen. Je nach Nutzungsintensität ist die Einarbeitungszeit umfangreicher, jedoch können Nutzende auf eine Reihe von unterstützenden Tools zurückgreifen, die den sehr technischen Umgang vereinfachen und leichter zugänglich machen. Aus den Erfahrungen, die an den Hochschulstandorten gemacht wurden und die in der AG GitLab der Landesinitiative fdm.nrw geteilt wurden, ist der Bedarf nach einer Sammlung gut vermittelbarer Einsatzszenarien von git und GitLab im FDM erwachsen und umgesetzt worden. Wie dort gezeigt wird, lassen sich über CI diverse FDM-Tools anschließen und in einem Gesamtablauf von Erzeugung über Analyse bis zur Publikation integrieren. Diese Sammlung soll zukünftig erweitert und in virtuellen Veranstaltungen eingesetzt werden, sodass sich der Einsatz in der FDM-Community weiterverbreitet.