Automatisiertes Deployment von Elektronischen Laborbüchern mit Ansible

Henning Timm[1], Anne Wittkamp[2] und Stefan Beyer[3]

[1]Research Data Services, Universitätsbibliothek, Universität Duisburg-Essen, Deutschland
[2]Stabsstelle eScience, Zentrum für Informations- und Mediendienste, Universität Duisburg-Essen, Deutschland
[3]Geschäftsbereich IT-Infrastruktur, Zentrum für Informations- und Mediendienste, Universität Duisburg-Essen, Deutschland

Die spezifischen Anforderungen verschiedener Disziplinen an Elektronische Laborbücher (ELB) lassen sich nicht mit einer einzelnen Software erfüllen. Auch ist die Verwaltung einer zentralen, institutionellen ELB-Instanz für viele Anwendungsfälle zu unflexibel. Die Installation und der Betrieb vieler paralleler Instanzen erzeugen jedoch einen erheblichen Mehraufwand für die betreibenden Institutionen. Dieser Mehraufwand kann durch die Verwendung von automatisiertem Deployment mit Hilfe einer Virtualisierungsumgebung und einer Konfigurations- und Automatisierungssoftware, wie z. B. Ansible reduziert werden.

1 Betriebskonzepte für Elektronische Laborbücher

Die Dokumentation von Forschungsprozessen in der Laborforschung erfolgt bislang zum überwiegenden Teil analog in papierbasierten Laborkladden oder in einer Mischform von papierbasierten Laborkladden („Laborjournal“) und digitaler Datenerfassung. Elektronische Laborbücher bilden diese papierbasierten Laborkladden in Form von Webanwendungen ab. Neben der schlichten Möglichkeit die Dokumentation der Versuche direkt in digitaler Form zu erzeugen, bieten diese ELB viele weitere Features. Im Sinne eines digitalen Forschungsdatenmanagements erleichtert die Nutzung eines ELB die direkte Verknüpfung von Rohdaten mit zusammenhängenden Metadaten, Workflows und prozessierten und analysierten Daten und macht Forschungsprozesse sowie deren Ergebnisse besser nachvollziehbar und nachnutzbar. Die Gefahr eines Informationsverlustes oder fehlerhafter Ergebnisse, zum Beispiel wenn durch Zwischenschritte eines Workflows auf nicht aktuelle Daten zurückgegriffen wird, kann so vermieden werden. Ebenso können Daten zwischen Kooperationspartner:innen meist in nachhaltiger Weise ausgetauscht und transferiert werden.

Letzteres ist durch die Implementierung als Webanwendung möglich, wodurch die Inhalte des ELB direkt zentral abgelegt werden und über das Setzen von Berechtigungen mit anderen Nutzern geteilt werden können.

Publiziert in: Vincent Heuveline, Nina Bisheh (Hg.): E-Science-Tage 2021. Share Your Research Data. Heidelberg: heiBOOKS, 2022.
DOI: https://doi.org/10.11588/heibooks.979.c13744 (CC BY-SA 4.0)

Im Moment ist der Markt verfügbarer ELB-Lösungen sehr umfangreich [1]. Die speziellen Anforderungen verschiedener Disziplinen an ELB – etwa in Bezug auf die Anbindung von Datenbanken, technischen Geräten aus den Laboren oder besondere Darstellungen z. B. von Molekülen – gehen weit auseinander. Dadurch reichen die ELB-Lösungen von generischen hin zu fachspezifischen Programmen, die wiederum kommerziell oder als Open Source-Lösung zur Verfügung stehen. Bei der Einführung eines ELB am Standort müssen daher die Bedarfe der Forschenden an die Software gegen die technische und personelle Umsetzbarkeit der betreibenden Einrichtungen abgewogen werden. Die Erfahrung der Autor:innen zeigt, dass eine zentrale ELB-Instanz meist nicht ausreichend ist, um die fachspezifischen Bedarfe hinreichend abzudecken. Resultierend werden derzeit (auch an ein und demselben Standort) verschiedene ELB parallel eingeführt.

Der nachhaltige, zentrale Betrieb von mehreren heterogenen ELB-Instanzen stellt lokale FDM-Infrastruktur vor große Herausforderungen, nicht nur bei der Frage nach Speichermöglichkeiten und dem Identity Management.

Jede neue ELB-Instanz erzeugt Aufwände zur initialen Inbetriebnahme, wie die für Installation und Konfiguration benötigte Zeit, und bindet Personal für regelmäßige Wartung und Updates. Die Erfahrung der Autor:innen aus der Inbetriebnahme von vier ELBs zeigt, dass die Installation eines ELB auf einer Virtuellen Maschine (VM) ca. einen Personentag benötigt. Einige Arbeitsschritte, wie die Vergabe eines SSL-Zertifikates, hängen dabei von externen Stellen ab. Außerdem steigt durch ein breites, passgenaues Angebot an ELBs der Wartungsaufwand erheblich. Jede Instanz muss unabhängig Sicherheitsupdates erhalten und neue Versionen der ELB-Software müssen installiert werden.

Unabhängig von der konkret verwendeten Software weisen die meisten ELBs die Struktur einer typischen Webanwendung auf: Ein für das ELB spezifisches Frontend wird über einen Webserver zur Verfügung gestellt und interagiert mit einer Datenbank. Durch diese Struktur ähneln sich bestimmte Installationsschritte von ELBs, was bei der Automatisierung der Installation ausgenutzt werden kann. Weiterhin werden ELBs häufig mit Hilfe einer Virtualisierungsumgebung auf VMs installiert, die auf Hardwareanforderungen des konkreten ELBs angepasst werden können.

2 Automatisierung mit Ansible

Durch ein automatisches Deployment von ELB-Instanzen mittels einer Konfigurationssoftware wie Ansible[2] kann der Arbeitsaufwand erheblich verringert werden [3].[1] Automatisiertes Deployment bezeichnet hierbei die Installation und Konfiguration der benötigten ELB-Software auf einer bestehenden VM. Mit Hilfe von Ansible werden die dafür benötigten Schritte durch sogenannte Ansible-Playbooks zentral gesteuert und abstrahiert für ein sogenanntes Inventory von VMs durchgeführt: Ausgehend von einem Ansible-Server, auf dem die Playbooks hinterlegt sind, greift ein spezieller Ansible-Nutzer automatisiert

[1] Es existieren weitere Softwarelösungen, wie z. B. Saltstack oder Puppet, die ebenfalls für diese Aufgabe genutzt werden könnten.

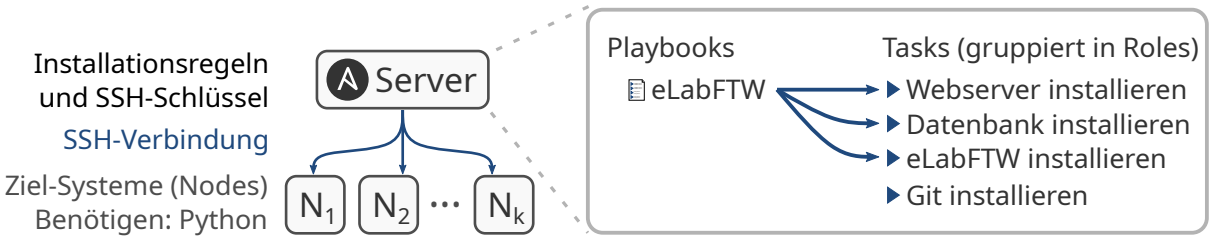

Abbildung 1: Beispielhafte Struktur eines Ansible-Servers mit k angeschlossenen VMs $N_1, .., N_k$ und einem Playbook zur Installation von eLabFTW. Die Roles (z. B. Webserver Installieren) beinhalten jeweils mehrere Tasks (Installation der Software, Konfiguration, ...).

per SSH auf die angeschlossenen Systeme zu und führt dort die für die jeweilige VM vorgesehenen Aufgaben durch. Die Zielsysteme werden vom Ansible-Server in einem oder mehreren Inventories verwaltet.

Playbooks werden in der weit verbreiteten Markup-Sprache YAML definiert, was eine leichte Nachnutzung ermöglicht. Sie enthalten Tasks, einzelne Installations- und Konfigurationsschritte wie z. B. die Installation einer Software mit Hilfe der Paketverwaltung, die in Roles für bestimmte Aufgaben zusammengefasst sind. Abbildung 1 zeigt beispielhaft die Struktur eines Playbook für die Installation des ELBs eLabFTW [4].

Installationen und sichere Konfigurationen (z. B. gehärtete Webserverkonfiguration, Firewalleinstellungen) müssen so nur einmal an zentraler Stelle erstellt, geprüft und aktualisiert werden. Soll eine neue Instanz eines bereits verwendeten ELBs in Betrieb genommen werden kann das bestehenden Playbook verwendet werden und es müssen nur Anpassungen, wie z. B. Passwörter, Namen und IP-Adressen, für die neue VM in das Inventory eingepflegt werden.

Im Gegensatz zu anderen Deploymentlösungen, wie z. B. dem Klonen eines Golden Image bei Erstellung einer VM, ermöglicht Ansible auch eine automatisieren Wartung der Systeme. So können Backups, System- und Softwareupdates ebenfalls zentral für alle angeschlossenen VMs ausgelöst werden. Da die angeschlossenen VMs mit dem selben Playbook installiert wurden und daher eine weitestgehend identische Konfiguration aufweisen reduziert sich der Wartungsaufwand für das betreibende Personal. Durch die zentrale Steuerung von Updates über den Ansible-Server werden ebenfalls die Einarbeitungszeiten in die jeweilige Software reduziert, was den Prozess sowohl beschleunigt als auch gegen Flüchtigkeitsfehler absichert. Zudem fungieren die gesammelten Playbooks als Dokumentation des Installationsprozesses und als zentraler Ort, an dem Besonderheiten der verwendeten Software notiert werden können.

Die gewonnene Zeitersparnis wird allerdings erkauft durch einen höheren Vorbereitungsaufwand, namentlich die Entwicklung (und Wartung) der Playbooks. Da die Entwicklung eines Playbooks nur einmalig anfällt und dessen Wartung für alle assoziierten Instanzen nur an einer Stelle (eben im Playbook) durchgeführt werden muss, amortisieren sich dieser Aufwand. Dieser Effekt wird noch verstärkt, wenn neu entwickelte Playbooks auf

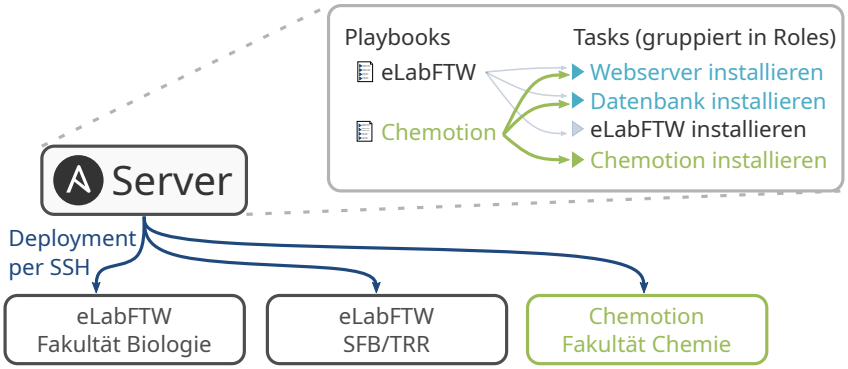

Abbildung 2: Fallbeispiel für die Entwicklung eines neuen Playbooks für die Software Chemotion, aufbauend auf einem existierenden Playbook für eLabFTW. In diesem Fall soll Chemotion mit der selben Webserver- und Datenbankkonfiguration wie eLabFTW betrieben werden, so dass die entsprechenden Roles nachgenutzt werden können.

bereits bestehende Roles und Tasks zurückgreifen können. Initial hat die Entwicklung des eLabFTW-Playbooks ca. 20 Stunden in Anspruch genommen und momentan werden neun Instanzen von eLabFTW mit diesem System betrieben. Die größte Zeitersparnis entsteht jedoch in Betrieb und Wartung durch die Möglichkeit Updates zentral ausgelöst, parallel und automatisiert für alle Instanzen durchzuführen. Der Wartungsaufwand, welcher vorher ca. 1 Stunde pro Monat pro Instanz betrug, konnte so auf 1 bis 2 Stunden pro Monat für alle neun Instanzen reduziert werden.

Bestehende Playbooks können aber auch als Grundlage für weitere Applikationen dienen. Bei ELBs die der oben beschriebenen Struktur von Frontend, Webserver und Datenbank folgen, aber auch bei verwandter Software können entwickelte Regeln für neue Playbooks nachgenutzt werden. Die Entwicklung neuer Playbooks (s. u.) benötigt aktuell nur noch 3 bis 6 Stunden, da große Teile existierender Playbooks nachgenutzt werden können.

In dem in Abbildung 2 dargestellten Fallbeispiel soll zu zwei bestehenden Instanzen von eLabFTW ein mit der Software Chemotion[5] realisiertes ELB hinzugefügt werden. Ein für Chemotion neu entwickeltes Playbook kann, abhängig von der angestrebten Konfiguration, z. B. die für eLabFTW entwickelten Roles für Webserver und Datenbank nachnutzen. Damit reduziert sich die Entwicklung des Playbooks auf die für Chemotion spezifischen Schritte.

3 Inter-Institutionelle Kollaboration

Eine Stärke der Open Source Software Ansible, im Gegensatz zu vergleichbarer Softwarealternativen, liegt in der geringen Einstiegshürde für den Betrieb des Servers, sowie für Entwicklung und Austausch von Playbooks. Playbooks können unabhängig von Inventories (die sicherheitskritische und interne Informationen beinhalten) leicht über

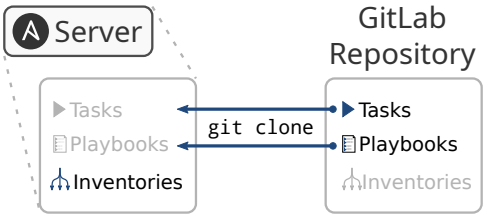

Abbildung 3: Beispiel für die Arbeit mit kollaborativ entwickelten Playbooks. Öffentliche Informationen, wie die verwendeten Playbooks, können über eine Plattform wie GitLab geteilt werden. Interne und sicherheitskritische Anpassungen, wie z. B. Inventories oder Anpassungen für bestimmte Systeme, können von der Institution intern verwaltet werden.

Software-Plattformen wie z. B. eine institutionelle GitLab-Instanz verwaltet und ausgetauscht werden. Auf dem Ansible-Server werden diese dann via Git synchronisiert und mit den Institutionsspezifischen Inventories verknüpft. Dabei muss besonders beachtet werden, dass das Playbook keine internen Informationen, wie z. B. Zugangsdaten, preis gibt. Im Fall der Autor:innen wurde das Playbook in Workshop-Form (z. T. mit externen Gästen) auf einem Testsystem entwickelt, so dass diese Abgrenzung von Beginn an eingehalten wurde. Das entwickelte eLabFTW-Playbook ist auf GitHub veröffentlicht (`https://github.com/RDS-UDE/eLabFTW-ansible-playbook`).

Dieses in Abbildung 3 illustrierte Vorgehen erlaubt die kollaborative Entwicklung im Austausch mit anderen Institutionen, ermöglicht aber ebenfalls die Verwaltung interner Komponenten. Soll z. B. eine neue FDM-Software getestet oder eingesetzt werden, für die bereits ein Playbook an einer anderen Institution entwickelt wurde, so kann dies den Aufwand der Inbetriebnahme deutlich reduzieren. In diesem Fall können dieses Playbook und die assoziierten Roles überprüft und (z. B. via Git) in die bestehende Ansible-Architektur eingebunden werden, so dass nur das Inventory angepasst werden muss. Durch eine gemeinsame Entwicklung mit Kooperationspartner:innen anderer Institutionen können so zum einen einheitliche Konfigurationen entwickelt werden, zum anderen erhöht die mit einer Nachnutzung einhergehende Überprüfung die Qualität des entwickelten Codes.

4 Fazit

Mit Hilfe von Ansible kann der erhöhte Installations- und Wartungsaufwand, der mit dem Betrieb individueller ELBs einher geht, reduziert werden. Durch ihre Struktur als Webanwendung sind ELBs besonders geeignet für dieses Vorgehen, da ein großer Teil der benötigten Installationsschritte Potential zur Nachnutzung in anderen Playbooks aufweist. Zusätzlich wird durch die Erstellung und Wartung von Playbooks eine lebendige Dokumentation der betriebenen Systeme geschaffen. Ein späterer Umstieg auf Automatisierungstechnologien mit anderem Fokus (z. B. Docker und Kubernetes) bleibt durch die Struktur von Ansible-Projekten ebenfalls möglich.

Die Verwendung einer Open Source Software mit niedriger Einstiegshürde ermöglicht die Nachnutzung von Tasks und Playbooks, sowohl intern – zur Entwicklung neuer Playbooks – als auch in Kooperation und Kollaboration mit externen Institutionen (z. B. organisiert über GitLab).

Acknowledgements

Die Rechte an Logos gehören den jeweiligen Rechteinhabers: Das verwendete Ansible-Logo ist abgeleitet von der Datei Ansible_logo.svg, erstellt durch den Wikimedia Commons Nutzer Vulphere, welche gemeinfrei zur Verfügung steht.

Literaturverzeichnis

[1] ZB MED (Hrsg.). 2020. *Elektronische Laborbücher im Kontext von Forschungsdatenmanagement und guter wissenschaftlicher Praxis – ein Wegweiser für die Lebenswissenschaften*, 2. aktualisierte und erweiterte Fassung, Köln, doi: `https://doi.org/10.4126/FRL01-006415715`.

[2] Red Hat, Inc. 2021. „Ansible is Simple IT Automation“. Abgerufen 14. Mai 2021 von: `https://www.ansible.com/`.

[3] Masek, Pavel, Martin Stusek, Jan Krejci, Krystof Zeman, Jiri Pokorny und Marek Kudlacek. 2018. „Unleashing Full Potential of Ansible Framework: University Labs Administration“. 22nd Conference of Open Innovations Association (FRUCT), 144-150, doi: `https://doi.org/10.23919/FRUCT.2018.8468270`.

[4] CARPi, Nicolas, Alexander Minges und Matthieu Piel. 2017. „eLabFTW: An open source laboratory notebook for research labs“. Journal of Open Source Software, 2(12), 146, doi: `https://doi.org/10.21105/joss.00146`.

[5] Tremouilhac, Pierre, An Nguyen, Yu-Chieh Huang, Serhii Kotov, Dominic Sebastian Lütjohann, Florian Hübsch, Nicole Jung und Stefan Bräse. 2017. „Chemotion ELN: an Open Source electronic lab note-book for chemists in academia“. J Cheminform 9, 54, doi: `https://doi.org/10.1186/s13321-017-0240-0`.