Journal of Biotechnology 341 (2021) 43–50

Contents lists available at ScienceDirect

Journal of Biotechnology

journal homepage: www.elsevier.com/locate/jbiotec

A local platform for user-friendly FAIR data management and reproducible analytics

Florian Wieser [a,1], Sarah Stryeck [b,c,1], Konrad Lang [b,c], Christoph Hahn [d], Gerhard G. Thallinger [e,i], Julia Feichtinger [f,i], Philipp Hack [g], Manfred Stepponat [g], Nirav Merchant [h], Stefanie Lindstaedt [b,c,**], Gustav Oberdorfer [a,i,*]

[a] Institute of Biochemistry, Graz University of Technology, 8010, Graz, Austria
[b] Institute for Interactive Systems and Data Science, Graz University of Technology, 8010, Graz, Austria
[c] Know-Center GmbH, 8010, Graz, Austria
[d] Institute of Biology, University of Graz, 8010, Graz, Austria
[e] Institute of Biomedical Informatics, Graz University of Technology, 8010, Graz, Austria
[f] Division of Cell Biology, Histology and Embryology, Gottfried Schatz Research Center, Medical University of Graz, 8010, Graz, Austria
[g] Central Information Technology, Graz University of Technology, 8010, Graz, Austria
[h] Data Science Institute, University of Arizona, BSRL 200 A, Tucson, AZ, 85721, United States
[i] BioTechMed-Graz, Austria

ARTICLE INFO

Keywords:
Cyberinfrastructure
Bioinformatics
Research data management
FAIR
Teaching
CyVerse

ABSTRACT

Collaborative research is common practice in modern life sciences. For most projects several researchers from multiple universities collaborate on a specific topic. Frequently, these research projects produce a wealth of data that requires central and secure storage, which should also allow for easy sharing among project participants. Only under best circumstances, this comes with minimal technical overhead for the researchers. Moreover, the need for data to be analyzed in a reproducible way often poses a challenge for researchers without a data science background and thus represents an overly time-consuming process. Here, we report on the integration of CyVerse Austria (CAT), a new cyberinfrastructure for a local community of life science researchers, and provide two examples how it can be used to facilitate FAIR data management and reproducible analytics for teaching and research. In particular, we describe in detail how CAT can be used (i) as a teaching platform with a defined software environment and data management/sharing possibilities, and (ii) to build a data analysis pipeline using the Docker technology tailored to the needs and interests of the researcher.

1. Introduction

With experiments nowadays generating a wealth of research data, the term 'Big data' has become a buzzword in life science. Thus, dealing with large amounts of data has become exceedingly challenging for life science researchers without a data science background. A substantial boost in handling these amounts of data came from establishing resources and methods for data management/archival as well as discipline-specific standardized data formats, to ensure reproducibility. However, there is an apparent lack of tools, which support life science researchers to efficiently use their data, specifically with respect to data analytics and collaborative research on large datasets. As a result, rendering research data FAIR (findable, accessible, interoperable and reusable) before and after publication represents an overly time-consuming process for researchers. CyVerse US (https://www.cyverse.org, 2021), an initiative from the University of Arizona, supports research-related processes from data generation, management, sharing and collaboration for data analytics and storage. All those processes are essential for following a FAIR approach to research data.

The initiative started with a strong community in life science and is now expanding to other disciplines, with several entities deploying the CyVerse infrastructure outside the US, such as CyVerse UK

* Corresponding author at: Institute of Biochemistry, Graz University of Technology, 8010, Graz, Austria.
** Corresponding author at: Institute for Interactive Systems and Data Science, Graz University of Technology, 8010, Graz, Austria.
E-mail addresses: slind@know-center.at (S. Lindstaedt), gustav.oberdorfer@tugraz.at (G. Oberdorfer).
[1] Equal contribution.

https://doi.org/10.1016/j.jbiotec.2021.08.004
Received 31 January 2021; Received in revised form 24 June 2021; Accepted 4 August 2021
Available online 13 August 2021

(https://cyverseuk.org, 2021) or initiatives in Australia (https://cyverse.org/Researchers-Explore-Creation-of-CyVerse-Australia, 2021). However, all those initiatives are still linked to CyVerse US (https://www.cyverse.org, 2021). We deployed an independent instance of CyVerse in Austria, which is currently used by three universities – Graz University of Technology (TUG), University of Graz and Medical University of Graz – as a shared research data platform (https://cyverse.tugraz.at, 2021; Lang et al., 2020). CyVerse Austria (CAT) introduces a new cyberinfrastructure (CI) for a local community of life science researchers and additionally provides a huge potential to serve researchers beyond the life science domain at various institutions within Austria. Moreover, this project sets a new basis to support collaborations and reproducible research between universities by (i) creating a distributed computational and data management architecture for FAIR research data, (ii) hosting relevant tools as Docker containers (Devisetty et al., 2016; https://www.docker.com, 2021) to ensure reproducible analytics and (iii) integrating researchers into a global community. Previously published work on CAT focused on the detailed technical setup of CAT (Lang et al., 2020). In this article, we elucidate CAT from the user perspective, illustrated by two step-by-step tutorials which should serve as protocols for researchers who intend to use CAT for their research or teaching. It is essential to understand the value of a CI without the need of having extensive knowledge in data/computer science or high-performance computing (HPC) infrastructure. Here, we mainly address researchers that need analytical tools for their research but do not have extensive *a priori* knowledge about the underlying technical requirements. Therefore, we highlight and describe in detail how CAT can be used (i) as a teaching platform with a defined software environment and data management/sharing possibilities, and (ii) to build a data analysis pipeline using the Docker technology tailored to the needs and interests of the researcher.

2. Methods

2.1. Implementation

TUG provides the core system for user management and the web front-end together with storage and computational hardware. Distributed storage is provided using the integrated Rule Oriented Data System (iRODS) technology (https://irods.org, 2021). Each participating institution integrates its own storage resources via the iRODS resource server. Since each of the institutions provides local storage devices, it is ensured that research data is stored at the corresponding institution – a legal requirement when analyzing personal data, patient data or data generated within industry related projects (Fig. 1).

Jobs for data analytics submitted in CAT are scheduled to the computing resources through HTCondor (https://research.cs.wisc.edu/htcondor, 2021; Litzkow et al., 1988; Thain et al., 2005). These can either be submitted to the central resources at TUG or to high performance computing (HPC) clusters of the other participating institutions. For submission of jobs to the HPC clusters, there are HTCondor transfer nodes deployed at the servers of each participating institution to connect with other schedulers such as SGE (https://arc.liv.ac.uk/trac/SGE, 2021) or Slurm (https://www.schedmd.com, 2021; Yoo et al., 2003). Jobs are submitted as Docker containers (https://www.docker.com, 2021) and converted to Singularity containers (Kurtzer et al., 2017) at the HPC clusters if required to adhere to security restrictions of the participating institutions.

The user sees all these services combined in the CAT Discovery Environment (DE). The DE offers a graphical user interface (GUI) to

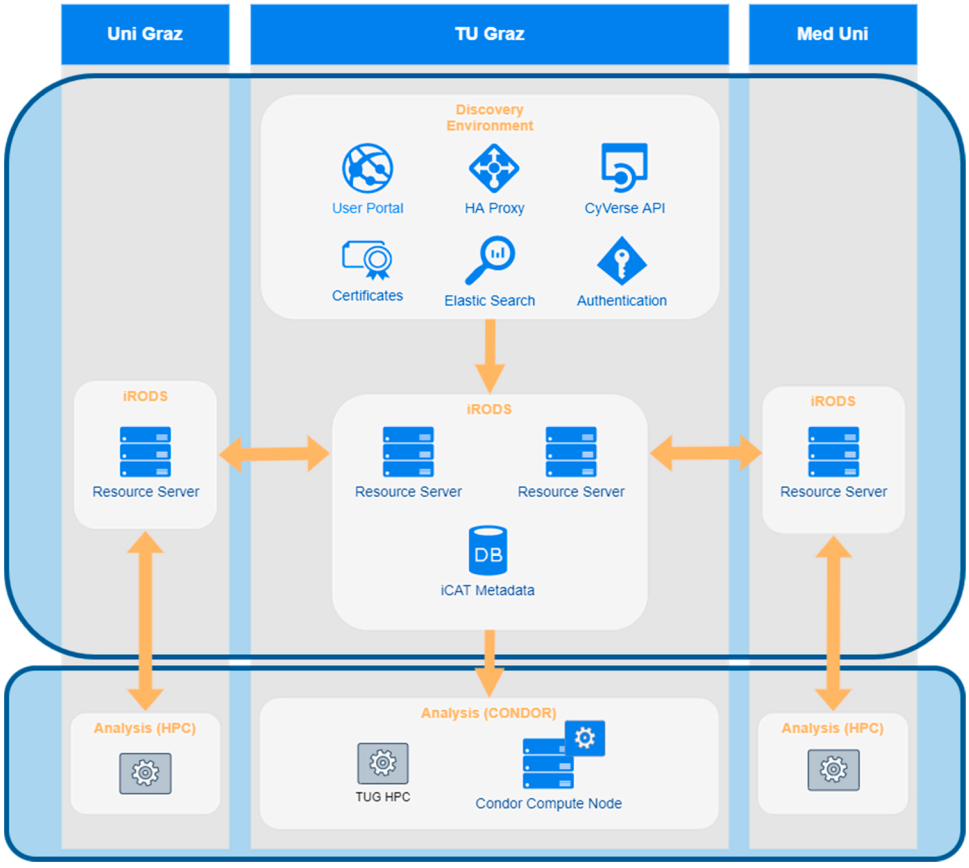

Fig. 1. Overview of the current hardware organization of CAT. Each participating institution has their own resource servers connected to the core system of TU Graz, which hosts the DE and the iCAT metadata catalogue. Institutions have access to both, their internal HPC hardware as well as the central HPC infrastructure at TU Graz.

present (i) the data in the folder structure with files managed in CAT, (ii) the Apps representing all available tools as Docker containers in CAT and (iii) the status of currently running, as well as completed jobs. Data analytics in CAT can either be run as a script in the background or by using a visual and interactive computing environment (VICE).

CAT is based on the open-source code provided by CyVerse US, however, specific adaptions had to be implemented to be adhere to university regulations. Thus, some of the CAT modules are deployed differently. For instance, CAT establishes the connection to HPC clusters at different institutions, whereas CyVerse originally was built to provide central computational power and connects the CI to the OpenScience Grid (Pordes et al., 2007). In addition, the user account management is different from its original implementation. Currently, user accounts are created by the CAT team. In the future, Keycloak (https://www.keycloak.org, 2021) will be used as Open Source Identity and Access Management tool to enable single sign-on for members of Austrian universities.

3. Results

3.1. Operation

CAT is a local, independent instance based on CyVerse US (*1*). University employees of the participating institutions can request a user account from the CAT team and connect to it through the university network or using a Virtual Private Network (VPN) client. All services are running in CAT, and there are no specific requirements for the underlying system used by the researcher, except the connection to the institutional network. Solely preparation of new containers has to be performed locally using Docker (Devisetty et al., 2016; https://www.docker.com, 2021), which serve as the basis for Apps designed in the DE.

3.2. Usability

In CAT, users interact mostly with the DE, which offers a GUI for intuitive research data management, data storage, data sharing, initiating and monitoring of computational processes, as well as accessing results of executed workflows and analyses (Fig. 2). CAT allows users to store and access their data in an easy and intuitive way using the CyVerse Data Store (DS) with the DE GUI. Here, a researcher stores files containing research data in folders and subfolders.

3.3. Documentation is key

Research in different disciplines suffers from the reproducibility crisis (Doleman et al., 2019; Fanelli, 2010; Ioannidis and Trikalinos, 2007). Therefore, there are top-down (Directorate, 2021) and bottom-up (Schönbrodt et al., 2021) movements supporting adequate research data management (RDM) and open science practices. Funding agencies and publishers demand RDM according to FAIR principles. In order to meet those requirements, it is essential to have the technical infrastructure in place.

Features in CAT that support all FAIR guiding principles are described below (letter/number pairs in this section correspond to (Wilkinson et al., 2016):

3.3.1. Findable

Unlike most repositories that only support data during and after publication, CAT supports users in generating FAIR data throughout the entire data life cycle. CAT is not intended as a long-term data archive, but it provides an API to transfer data to long-term repositories (e.g. InvenioRDM). Published data with globally unique and persistent identifiers (also known as PIDs), can be linked to CAT datasets (F1, F3). All data in CAT are described with rich metadata (F2). Additional scientific metadata, based on community input and standards, are required for certain data types (F2). CAT data are indexed and discoverable through the CyVerse instance of ElasticSearch (F4).

3.3.2. Accessible

Data and metadata in CAT are retrievable through standard communication protocols with other members (A1.2). CAT provides pipelines to publish to canonical repositories such as NCBI, as well as institutional repositories (e.g. InvenioRDM) to make (meta)data retrievable by their PID through open, free and universally implementable protocols (A1.1). Metadata are stored in the iCAT metadata catalogue and can be kept even if data are no longer available (A2) to provide a high degree of FAIRness of the research.

3.3.3. Interoperable

Data in CAT must be in non-proprietary formats that are readable by widely accessible software (I1). Metadata for published datasets are available as download in JSON, with citations as BibTeX and EndNote (I1, I2). Vocabularies and standards are always evolving, especially for

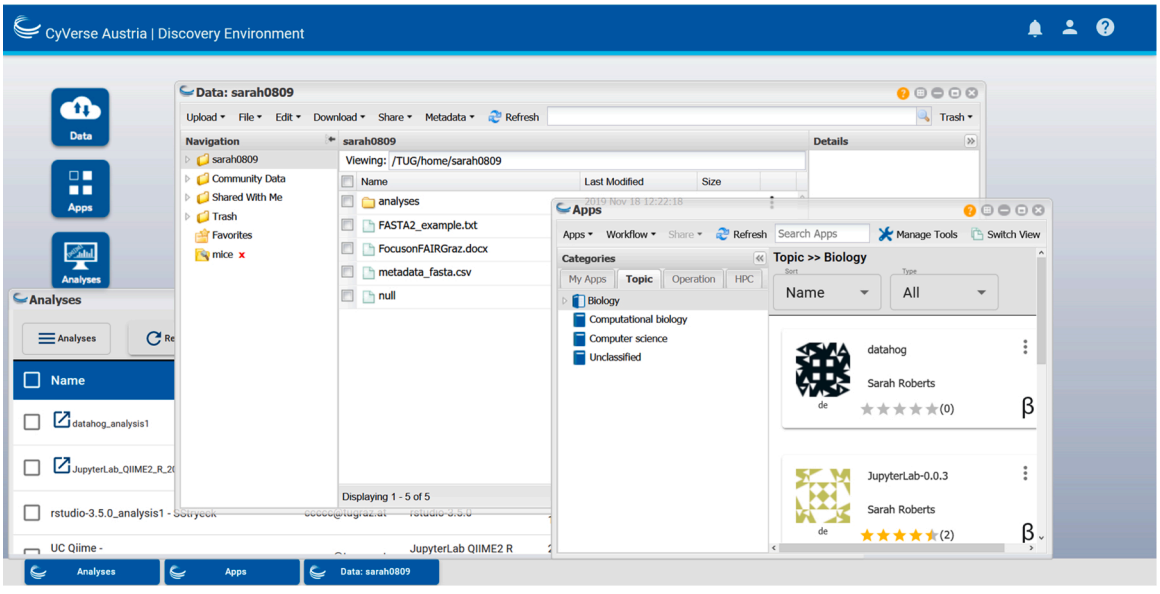

Fig. 2. Layout of the DE in CAT. On the left the user can open and switch between the three main windows, which encompass a data browser, list of available apps, as well as an analysis window. All features of CAT can be accessed via this interface.

newer data types, so CAT continues to work with communities on specification and adoption. Using data attributes such as relatedIdentifer, (meta)data in CAT can be linked to other data (I3). The CAT metadata API is engineered to support links among metadata elements (i.e. metadata graphs) using formal ontological and RDF properties and allows the use of data models (I3). Data models provide a way to relate data elements to one another and are essential for managing large, complex datasets.

3.3.4. Reusable

To meet the FAIR reusability criterion, (meta)data must be richly described with accurate and relevant attributes including a clear and accessible data usage license and detailed information about its origin, following domain-relevant community standards. All of this CAT currently does for published data (R1.1–1.3). Dublin Core (https://www.dublincore.org/specifications/dublin-core/dcmi-terms, 2021) and DataCite (Martone, 2014) do not truly support reusability due to their focus on publication rather than science. Therefore, CAT adopts standards developed by scientific communities and works closely with communities that want to encourage rich metadata usage.

In addition to FAIR data principles, CAT supports reproducible science by the deployment of Docker containers and through good data management practices, such as tracking data origin through analysis steps, recording results in standardized formats, and providing access to scripts, runs, and results.

3.4. Reproducible research with containers

A Docker container is a standard execution unit that packages code and all its dependencies, thereby making it easily portable to other computing environments. However, on HPC clusters Singularity containers are preferred, as they can be executed without root privileges. Therefore, root privilege escalation is reduced. HPC environments are typically multi-user systems, where users should only have access to their own data. For all practical purposes, Docker requires superuser privileges, which does not adhere to the security requirements of most HPC clusters. To this end, Docker containers can be easily converted to Singularity containers, which means that dockerized tools stored in the DE can be send to the HPC cluster, converted into Singularity format and run on the HPC cluster.

3.5. Life science towards data science

Life science research is becoming increasingly ‘data heavy’, with analysis workflows often exceeding the computational capabilities of regular laptops or desktop computers. Institutions therefore provide HPC clusters where researchers can request access and perform their respective computation. CAT enables researchers to directly submit a computing job to the HPC clusters by starting an App. As CAT users are assigned to specific computing resources, their job will be directly submitted to the appropriate HPC and the output will be transferred back to the CAT data system.

3.6. Collaboration enables research

For efficient collaboration, it is essential that researchers have tools to share their research data. CAT enables sharing by providing distributed storage at different research institutions. However, for the user it does not matter where the data is stored. All data that is owned by or shared with the user, potentially via multiple separate storage servers, will be visible through the GUI of the CAT DE. Thus, researchers using CAT adhere to institutional regulations (https://www.forschungsdaten.info/fdm-im-deutschsprachigen-raum/oesterreich/fdm-policies, 2021). In the CAT DE, a user can assign read-, write- or ownership permissions to collaboration partners/groups for individual files, folders or datasets. Data can also be kept invisible for all users but the owner to adhere to data security guidelines.

4. CAT for teaching

4.1. Preparation prior to courses

University courses often include hands-on exercises, where the students have to use a specific software package to perform certain tasks or answer questions. The exercises often rely on a specific version of the software to function properly. Extensions of the software package may be necessary to perform a task, where also a certain release is required to yield reproducible results. In short, setting up the environment for computational exercises is often far from trivial and requires expertise, frequently beyond the level of early-stage students. There are basically three options to provide the students with a defined software environment for a course: (i) give detailed instructions on how the required version of the software package and its extensions are to be installed on the student’s computer; (ii) hold the course in a computer lab with its defined hardware and software environment; and (iii) provide the defined software environment as web-accessible service. Option (i) is feasible in principle, but subtle differences caused by the used operating system or extensive hardware requirements may prevent a successful setup of the course’s software environment. This is especially true for software packages with a short release cycle. Option (ii) is the preferred one when courses are held face to face, as the instructors have full control over the hard- and software environment and can be supported by the university’s IT department. However, if a course has to be held virtually, because participants are at different locations or physical contact is prohibited like in the COVID-19 pandemic, option (iii) is the most suitable.

In summer 2020, we offered two courses “Biostatistics and R” and “Applied Bioinformatics” to interested PhD students from all three universities in Graz in the context of the BioTechMed-Graz initiative (https://biotechmedgraz.at). Both courses require a defined R-environment (R Development Core Team, 2019) to allow the students to perform their tasks, where the bioinformatics course depends heavily on additional R-packages and requires significant hardware resources. We used the CAT infrastructure to provide the students with a defined software environment as an R-Studio (RStudio Team, 2020) instance.

Prior to the course, we configured a Docker container, fully equipped to serve as a visual and interactive computing environment, for the course participants via CAT. Our container built on an Rstudio server image, as provided by the Rocker project (Boettiger and Eddelbuettel, 2017; https://www.rocker-project.org, 2021), and was reconfigured/extended with nginx and iCommands to be compatible with the HTCondor and Kubernetes orchestration implemented in CyVerse (Swetnam, 2019) (see also https://github.com/cyverse-vice/rstudio-base). The resulting container was further extended with the particular libraries and R-packages required for the course (the Dockerfile detailing the setup can be found in the Supplementary Materials).

Students received a link from the help desk for direct access to a running RStudio instance, which was initiated beforehand by the platform team using the Docker container available in CAT.

4.2. Procedure during the course

To avoid students having to familiarize themselves with CAT as well as accessing and starting RStudio on their own, provisioning of applications was done by providing a simple URL. This allowed students to access RStudio using a web browser, which immensely simplified the process for the students, and furthermore ensured that all students worked with the same tool versions and were not restricted by certain hardware requirements or underlying dependencies. We used two strategies of data sharing to provide lecture notes, report templates and input files for the exercises: (i) students were provided with files externally (downloadable on a website with user-restricted access) and

uploaded them on their own using the 'files' pane of the RStudio instance; and (ii) we established common data access directly through CAT. Data was synchronized with each student's container automatically as soon as the lecturer added data needed for exercises in the DE. The students could access the data directly in R, as well as download it using the 'files' pane, if required. The latter facilitated simple data access for the students and proved to be particularly convenient for sharing large data files. During the course the students used RStudio in CAT to create their own scripts. Thereby the students could perform diverse analyses of varying complexity in R, ranging from the use of descriptive and inferential statistics to bioinformatics analyses such as a basic RNA-seq analysis workflow, using the R-packages Rsubread (Liao et al., 2019) and limma (Ritchie et al., 2015). The hardware resources available to each student were adjusted to meet the increasing demands of the given examples. The support team defined the resource limits for the containers used by the students and raised them upon request. The use of HPC resources allowed to perform more sophisticated and computationally intensive analyses such as the alignment of sequencing reads to a reference genome using the R-package Rsubread (Liao et al., 2019). Interaction between the students and lecturers was facilitated using an online meeting tool that allowed for breakout rooms. This enabled the students to work in teams as well as to ask questions via (video) chat and screen sharing. Finally, the students completed a report based on a provided template, which was handed in at the end of each unit.

4.3. Technical support during the course

To provide a unified way to communicate between students (users) and the support team, we decided to implement a communication channel with two access points and a unified back-end to handle such requests: (i) direct emails to a dedicated address, which are handled by the support team; (ii) filling out a contact form available on the website, which requires specifying the user's email address to effectively start communicating. Our choice for a unified back-end was Zammad (https://www.zammad.org, 2021), a web-based, open source and freely available user support and ticketing system.

There are simpler ways to implement the communication and support channel for web-forms and email entry points, but with the plans to integrate a direct chat function and other social media channels into the CAT platform in the future, Zammad became the tool of choice. Other tools are either focused on issue tracking (https://otrs.com, 2021; https://www.intercom.com, 2021) or not available as open source and free solutions. Furthermore, cloud-based solutions may also be hosted outside the EU where General Data Protection Regulation issues need to be considered.

5. CAT for reproducible analytics

5.1. Integration of an automated workflow into the CAT DE

Multi-staged computational pipelines where executables not only depend on the computational output of the preceding process, but also on a specific version, are typical in bioinformatics (Devisetty et al., 2016). With the possibility to implement multi-staged automated workflows in the DE and the direct access to high-performance computing resources, CAT provides a well-suited infrastructure for complex pipelines and computationally demanding tasks. We therefore chose the integration of a protein structure prediction pipeline as the prototypical example of a user-friendly automated workflow. In particular, we integrated the *ab initio* structure prediction protocol from the software package Rosetta (Leaver-Fay et al., 2011; Rohl et al., 2004).

The *ab initio* structure prediction protocol (Bender et al., 2016; Leman et al., 2020) attempts to solve the common problem of finding the lowest energy protein structure for a given amino-acid input sequence (Huang et al., 2016). It is frequently applied to proteins for which no homologous structure is available, as well as a self-consistency check after a protein sequence design task (Leman et al., 2020; Boyken et al., 2016; Huang et al., 2014). Fig. 3 shows a workflow diagram of the entire *ab initio* pipeline.

5.2. Workflow description

The pipeline is initiated by predicting a protein's sequence profile and secondary structure by using the algorithms PSIBLAST and PISPRED (Altschul et al., 1997; Jones, 1999). These algorithms take the amino acid sequence in FASTA format as input. Using the information of the sequence profile and the secondary structure, a protocol in Rosetta collects fragment sets from known protein structures for each position of the amino acid query sequence. These fragments typically range in the size of 3, 6 or 9 amino acids and include the bond angles of the backbone and/or the side chain atoms (Gront et al., 2011). Guided by an energy function and sampling via a Monte Carlo search (Leaver-Fay et al., 2011; Rohl et al., 2004), the actual *ab initio* protocol assembles the fragments in an attempt to find an energy minimum, which is considered as final prediction, presumably resembling the native state of the protein (Baker and Sali, 2001).

In order to identify the native state, a vast number of structure conformations have to be sampled which makes these kinds of algorithms in general computationally very demanding. Thus, production runs, during which 10.000–100.000 structures are computed, are usually performed on highly parallel, HPC systems (Kuhlman and Bradley, 2019; Raman et al., 2008).

During the last decade efforts have been made to make Rosetta accessible to a broader range of scientists, and particularly to users without a software-engineering related background. Hence two high-level interfaces called RosettaScripts and PyRosetta have been implemented (Lyskov et al., 2013). The latter is a Python-based implementation of Rosetta and allows to write custom structure prediction and design protocols either interactively using iPython, or script-based using Python scripting. An integration of PyRosetta into the CAT DE is intended in the future (Ford et al., 2020).

RosettaScripts on the other hand provides protocol-level access to almost all of Rosetta's modeling features with the aid of an XML-like language interface in which the user defines a set of RosettaScripts objects and their order of execution (Bender et al., 2016; Fleishman et al., 2011). These XML scripts are straight forward to write without having to know the underlying C++ code base. Moreover, they are easily ported to other systems that support Rosetta and make recompilation of the Rosetta source code obsolete (Fleishman et al., 2011). Hence we decided to integrate RosettaScripts as the second tool into the CAT DE. All Dockerfiles and wrapper scripts implemented within this project are available online at https://github.com/FlorianWieser1/rosetta-at-cyverse.

5.3. Generating docker containers

All containers created in this project are based on the current Ubuntu Long Term Support version 20.04 (Supplementary Materials, sample Dockerfile 2). Each Rosetta Docker container consists of one or more statically pre-compiled Rosetta executables (Supplementary Materials, sample Dockerfile 2, line 11), a wrapper start script (Supplementary Materials, sample Dockerfile 2, line 12) and the Rosetta main database. In order to keep the containers small, the Rosetta database is stored externally in the CAT data store. If the container is executed in the DE, the database will be mounted with the HTCondor node's working directory alongside with the other input files by using a combination of the docker run flags -v and -w (Devisetty et al., 2016). The -v flag (for volume) bind-mounts the Rosetta database to the container at runtime. Two other options that are especially useful to test the container's functionality offline on a computer are the -i (for interactive) and -t (for tty) flags. In combination with the 'entrypoint' flag they will start the container in manual mode, where the user has access to the

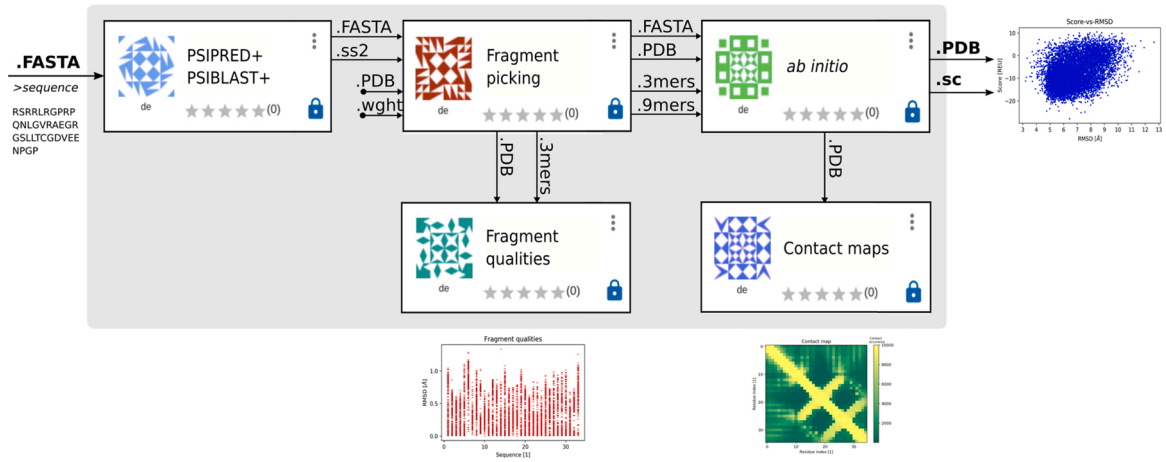

Fig. 3. Workflow diagram of the ab initio pipeline. Each rectangle represents an individual version-controlled Docker container. Continuous arrows indicate files transferred between containers, while arrows starting with a dot denote files which have to be submitted by the user. Containers in the first row are responsible for the actual protein structure prediction (see text for description). Containers in the second row compute quality metrics and generate charts.

command-line inside the container. A full command line example can be found in the Supplementary Materials.

In the command-line based Rosetta protocols the colon character (':') is extensively used within the execution parameters. Unfortunately, in the current version of the CAT DE (version 2.16.0) the colon is, due to security reasons, one of the forbidden special characters. This limitation was overcome by utilizing wrapper start scripts as entry points (see example code in Supplementary Materials).

Options of Rosetta executables can also be read from a text file. These text files, usually called 'flags' or 'options'-files are widely used in command-line based analyses, as the parameters for such computations are thereby inherently saved for re-use or sharing. If the wrapper script finds a file named 'flags' or 'options' (Supplementary Materials, sample Dockerfile 2, line 13 and 18) in the current working directory, the flags file will be passed as command line argument when executing the Rosetta protocol. If no such file is provided, it will pass the command line parameters provided in the launch menu of the application. In this case, the '-parser:protocol' flag will be appended, as this argument cannot be set in the DE due to the reasons mentioned above. Additionally, the wrapper script allows to run more than one executable at once and also handles the post-processing of output files, necessary for file transfer between Docker containers within workflows (see next Section). Furthermore, it is easy to debug a failed job in the CAT DE: Passing the output of basic bash commands such as "ls" or "pwd" into a textfile while running the container on CAT, will provide useful information such as the name of the Condor's current working directory or list all files in the working directory.

The CAT DE offers the possibility to assemble the individual Docker containers into concatenated workflows. In these workflows, the computational output of a preceding container is automatically passed as input to the subsequent one. The user has to solely provide the input files that are not automatically transferred (see arrows with preceding dot in Fig. 1). Workflows are convenient for comprehensive bioinformatics tasks, since once they are established, they can be easily adapted, re-used or shared with collaborators.

5.4. Setting up workflows in CAT

Workflows are set up by following the instructions under 'Workflow' – 'Create New…' in the DE. After providing general information, such as the workflow name and a description, the desired apps are added and arranged in the appropriate order of execution. Finally, the user has to define which output files will serve as input files for the subsequent computing step. Note that output files can only be forwarded and used as input files if they are labeled as such in advance in the corresponding apps. All passable output files in our *ab initio* pipeline are named 'outfile' followed by the appropriate file extension and were set not to be displayed in the launch menu of the application. In this way, computational output files predestined for transfer to the next container can easily be distinguished from the actual output of interest of the current container. In order to retain meaningful file names, computational outputs are copied to a results folder before renaming them to 'outfile' (Supplementary Materials, example script, lines 23–26) by the wrapper script. This has the beneficial side-effect that all intermediate computational results are stored in separate folders. By default, all outputs would be just left in the HTCondor's working directory (de-app-work), which would quickly become confusing, as in total more than 15 files are produced during one complete *ab initio* workflow run. Additionally, a clean-up container was integrated at the very end of the pipeline. It solely includes a bash script which will remove all files transferred between the containers containing the string 'outfile'.

6. Discussion

We introduced CAT, a new local CI for collaborative data research and simple data sharing for three universities in Austria. The infrastructure is set up in a fashion that guarantees data is handled according to FAIR principles. The platform has been broadly accepted by all participating institutions and fosters a fruitful environment for data management and analysis. In future, life science researchers need to intensify and build new collaborations with computer and data scientists. CAT is an example of a platform that can facilitate such collaborations, especially for highly interdisciplinary research. In general, CAT is composed of microservices, which makes it fully customizable depending on the requirements of users.

In our setup, we focused on three main areas: First, CAT makes it possible for new users to make efficient use of HPC resources without the need to have in-depth knowledge about high performance computing.

Second, we showed that CAT can easily be used for teaching purposes. We used the CAT environment to provide students with a defined software environment and sufficient hardware resources to hold two summer-school courses in lifescience data analysis using R. However, setup and test of the used R-Studio instance proved somewhat demanding, especially the installation of additional R-packages caused some difficulties. All packages had to be installed from source in the Linux-based Docker container, which often required installation of additional Linux packages to allow a successful compilation. This is in contrast to the Windows-based computer labs at the university, where

self-contained binary packages are installed, which do not require installation of operating system components. A very important aspect in the setup of the environment is a comprehensive test using the exemplary solution R-scripts to avoid any problems during the lecture. Nevertheless, the R-Studio instance within CAT provided an excellent option for our hands-on courses in the online setting. In combination with an open-source web-conferencing application it allowed us to cover all requirements for a successful virtual implementation of the courses. All data needed for the exercises could be stored in CAT and accessed directly from R-Studio by the students without the need to upload any additional data. This central data management system simplified data sharing between the lecturers and the students, who in this case were affiliated with different universities.

And third, we showed that CAT provides easy access to complex analysis pipelines that can otherwise be cumbersome to set up locally. The advantages of establishing a multi-staged workflow in the CAT DE for users include that (i) researchers with no command line experience have now access to a complex analytical pipeline via a GUI, (ii) the implemented analytics pipeline is preserved and allows reproducible analyses, (iii) researchers developing the pipeline become part of the global CyVerse community, which has a strong background in life science research and analytics, and (iv) other researchers can use the same pipeline and support/troubleshooting is simplified, since all input and output data are available via the iRODS resource server. This proves the value of a CI for collaborative and reproducible research.

In addition to the above-mentioned examples, we also scheduled regular user meetings and trainings to foster communication and exchange within the CAT community and across different universities.

7. Conclusion

CAT has been deployed in Graz, Austria to connect universities to improve collaborations and data sharing in this local research community. It connects three different institutions in Graz and serves life science researchers. The broader vision is to render CAT as a national platform for researchers to facilitate and implement interdisciplinary and collaborative research projects, strengthening research and FAIR data handling in Austria and abroad.

Apart from setting up the technical infrastructure, which enables data management and analytics as a service, it is equally important to ensure that the platform is indeed used and accepted. Therefore, it is essential to consider institutional culture with respect to data handling. In fact, this can mean that some aspects of how data is handled at the respective institution needs to change. This is, however, not an easy task but can, in our experience, be accelerated if appropriate training is offered regularly and documentation material is distributed widely. In addition, users have to be brought together to ensure that a community is built. This community will enhance the visibility of the platform and drive use case development as well as reinforce usage of built-in tools and workflows. As a result, the barrier for researchers to work with such a platform is lowered. Here we showcased examples of how such an infrastructure can benefit research collaborations and teaching efforts at the university level and provide a compact, yet comprehensive resource for building similar use cases elsewhere. Specific examples were given via two use cases for (i) data management and (ii) data analytics using CAT and an argument for the necessity and a net-value gain that a collaborative CI provides for researchers was made. In the future, we will further develop the platform according to the needs of our users.

8. Author contributions

FW, SS, KL, CH, GH, JF, SL and GO designed the research; FW, SS, CH, GGT and JF developed tools in CAT and conceptualized the use-cases; PH, MS, KL and SL helped with technical setup; NM, advised and help with the implementation of CAT; CH, GGT and JF developed and implemented all tools in CAT for use-case #1; FW and GO worked on use-case #2; FW developed and implemented all tools and containers for use-case #2; FW, SS, KL, CH, GGT, JF, NM and GO wrote the manuscript; all authors discussed the results and commented on the manuscript.

Software availability

CAT is based on CyVerse US and derived from the source code available at GitHub (https://github.com/cyverse), where specific implementations for CAT can be found in the CyVerse Austria Github Repository (https://github.com/cyverse-at). The platform is built using Google Web Toolkit (http://www.gwtproject.org/) for the user interface with an architecture of microservices deployed on a Kubernetes cluster to facilitate scalability and maintainability.

The code for DE of CyVerse is available on GitHub (https://github.com/cyverse-de) with specific implementations for CAT to be found at (https://github.com/cyverse-at).

Declaration of Competing Interest

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Acknowledgements

This research was funded by the Austrian infrastructure program 2016/2017, Bundesministerium für Bildung, Wissenschaft und Forschung Austria, BioTechMed/Graz Hochschulraum-Strukturmittel ‘Integriertes Datenmanagement’. The project was supported by Digitale TU Graz (Graz University of Technology). F.W. and G.O were supported by an ERC-StG (802217, HelixMold). J.F. was supported by a grant from the Austrian Science Fund (FWF): T923-B26.

Appendix A. Supplementary data

Supplementary material related to this article can be found, in the online version, at doi:https://doi.org/10.1016/j.jbiotec.2021.08.004.

References

Altschul, S.F., et al., 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 25, 3389–3402.
https://www.zammad.org, Accessed 22 June 2021.
https://www.cyverse.org, Accessed 22 June 2021.
https://cyverseuk.org, Accessed 22 June 2021.
https://cyverse.org/Researchers-Explore-Creation-of-CyVerse-Australia, Accessed 22 June 2021.
https://cyverse.tugraz.at, Accessed 22 June 2021.
https://www.docker.com, Accessed 22 June 2021.
https://irods.org, Accessed 22 June 2021.
https://research.cs.wisc.edu/htcondor/, Accessed 22 June 2021.
https://arc.liv.ac.uk/trac/SGE, Accessed 22 June 2021.
https://www.schedmd.com/, Accessed 22 June 2021.
https://www.keycloak.org/, Accessed 22 June 2021.
https://www.forschungsdaten.info/fdm-im-deutschsprachigen-raum/oesterreich/fdm-policies/, Accessed 22 June 2021.
https://www.dublincore.org/specifications/dublin-core/dcmi-terms/, Accessed 22 June 2021.
https://www.rocker-project.org/, Accessed 22 June 2021.
Baker, D., Sali, A., 2001. Protein structure prediction and structural genomics. Science 294, 93–96.
Bender, B.J., et al., 2016. Protocols for Molecular Modeling with Rosetta3 and RosettaScripts. Biochemistry 55, 4748–4763.
Boettiger, C., Eddelbuettel, D., 2017. An introduction to Rocker: Docker containers for R. R Journal 9, 527–536.
Boyken, S.E., et al., 2016. De novo design of protein homo-oligomers with modular hydrogen-bond network-mediated specificity. Science 352, 680–687.
Devisetty, U.K., Kennedy, K., Sarando, P., Merchant, N., Lyons, E., 2016. Bringing your tools to CyVerse Discovery Environment using Docker. F1000Res.
Doleman, B., Williams, J.P., Lund, J., 2019. Why most published meta-analysis findings are false. Tech. Coloproctol. 23, 925–928.

Fanelli, D., 2010. Positive" results increase down the hierarchy of the sciences. PLoS One 5.
Fleishman, S.J., et al., 2011. Rosettascripts: A scripting language interface to the Rosetta Macromolecular modeling suite. PLoS One 6.
Ford, A.S., Weitzner, B.D., Bahl, C.D., 2020. Integration of the Rosetta suite with the python software stack via reproducible packaging and core programming interfaces for distributed simulation. Protein Sci. 29, 43–51.
Gront, D., Kulp, D.W., Vernon, R.M., Strauss, C.E.M., Baker, D., 2011. Generalized fragment picking in rosetta: Design, protocols and applications. PLoS One 6.
Huang, P.S., et al., 2014. High thermodynamic stability of parametrically designed helical bundles. Science 346, 481–485.
Huang, P.S., Boyken, S.E., Baker, D., 2016. The coming of age of de novo protein design. Nature 537, 320–327.
Ioannidis, J.P.A., Trikalinos, T.A., 2007. An exploratory test for an excess of significant findings. Clin. Trials 4, 245–253.
Jones, D.T., 1999. Protein secondary structure prediction based on position-specific scoring matrices. J. Mol. Biol. 292, 195–202.
Kuhlman, B., Bradley, P., 2019. Advances in protein structure prediction and design. Nat. Rev. Mol. Cell Biol. 20, 681–697.
Kurtzer, G.M., Sochat, V., Bauer, M.W., 2017. Singularity: Scientific containers for mobility of compute. PLoS One 12.
Lang, K., et al., 2020. CyVerse Austria—A Local, Collaborative Cyberinfrastructure. Math. Comput. Appl. 25.
Leaver-Fay, A., et al., 2011. Meth. Enzymol. 487, 545–574.
Leman, J.K., et al., 2020. Macromolecular modeling and design in Rosetta: recent methods and frameworks. Nat. Methods 17, 665–680.
Liao, Y., Smyth, G.K., Shi, W., 2019. The R package Rsubread is easier, faster, cheaper and better for alignment and quantification of RNA sequencing reads. Nucleic Acids Res. 47.
Litzkow, M.J., Livny, M., Mutka, M.W., 1988. Proceedings - International Conference on Distributed Computing Systems, vol. 8, pp. 104–111.
Lyskov, S., et al., 2013. Serverification of Molecular Modeling Applications: The Rosetta Online Server That Includes Everyone (ROSIE). PLoS One 8.
Martone, M. (Ed.), 2014. Data Citation Synthesis Group, Joint Declaration of Data Citation Principles. FORCE11, San Diego CA.
Pordes, R., et al., 2007. J. Phys. Conf. Ser. 78.
R Development Core Team, 2019. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.
Raman, S., Qian, B., Baker, D., Walker, R.C., 2008. Advances in Rosetta protein structure prediction on massively parallel systems. Ibm J. Res. Dev. 52, 7–18.
Ritchie, M.E., et al., 2015. Limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res. 43, e47.
Rohl, C.A., Strauss, C.E.M., Misura, K.M.S., Baker, D., 2004. Meth. Enzymol. 383, 66–93.
RStudio Team, 2020. RStudio: Integrated Development for R. RStudio. PBC, Boston, MA.
Schönbrodt, F., et al., 2021. Netzwerk Der Open-Science-Initiativen (NOSI). OSF 22. Jan. 2021. Web.
Swetnam, T.L., 2019. cyverse-vice/rstudio-base: First Release (Version v0.1). Zenodo.
Thain, D., Tannenbaum, T., Livny, M., 2005. Distributed computing in practice: the Condor experience. Concurr. Comput. Pract. Exp. 17, 323–356.
Wilkinson, M.D., et al., 2016. Comment: the FAIR guiding Principles for scientific data management and stewardship. Sci. Data 3.
Yoo, A.B., Jette, M.A., Grondona, M., 2003. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 2862, pp. 44–60.
Directorate-General for Research and Innovation (European Commission), Options for Strengthening Responsible Research and Innovation, 2013. Publications Office of the EU.
https://otrs.com/, Accessed 22 June 2021.
https://www.intercom.com, Accessed 22 June 2021.