Containerization of Galaxy Workflows increases Reproducibility

Felix Bartusch, Maximilian Hanussek and Jens Krüger

High-Performance and Cloud Computing Group, IT Center, University of Tübingen

***Abstract*—Published scientific findings supported by computational experiments should be reproducible by following the methodical description in the publication. Because this is often not the case, we present a method to containerize Galaxy workflows within Singularity containers. These containers are executable such that other researchers can reproduce computational experiments or run their own experiments using the containerized workflows.**

I. Introduction

Scientific findings in fields like bioinformatics usually originate from processing primary data with computational methods to get results that agree or disagree with the initial assumption. The results are then published to make other researchers aware of the new findings and maybe new experimental methods. The applied methods are documented in the 'Methods' section of the paper and should ensure the reproducibility of the presented findings by other researchers. The content of this section differs from field to field but are often standardized. Examples for such standards in different fields are MIASE [1] for simulation experiments or MIAME [2] for microarray experiments. For computational methods it is consensus to state at least the used program version. Some publications state checklists of most important information needed to reproduce computational experiments [3].

But often the reproducibility is not guaranteed although the applied methods are described in detail in the corresponding publication [4], [5], [6]. Among others, the reason for irreproducibility is an insufficient description of used software, parameters and the data-processing workflow topology [6].

Scientific workflow engines like Galaxy [7], [8], [9], Taverna [10] and Knime [11] or portals like MoSGrid [12], [13] solve these problems by describing the computational workflow explicitly. In case of the mentioned workflow engines, the used software, parameters, and the workflow topology are described in a single file that can also be shared with other researchers via platforms like MyExperiment [14].

But also the use of workflow engines do not ensure full reproducibility because the shared workflow decays over time. A study based on a subset of Taverna workflows shared on the MyExperiment platform between the years 2007 and 2012 showed that around 80% of the workflows are not executable or do not produce the promised results any more [15]. At the time of the study the workflows had a maximal age of 5 years. The reasons for workflow decay are volatile third-party resources, missing example data, missing execution environment, and an insufficient description of the workflow.

Containerization techniques improved the mobility and usability of computations in the last years by encapsulating an operating system together with software and data. The most widely used containerization technique is Docker [16], but other techniques like Singularity [17] became more popular. Containerization of software increases reproducibility enormously [18]. One specific example is a Docker container providing a complex software stack for the simulation of spectroscopic fragmentation of small molecules with QCEIMS. The containerized software stack was used by UNICORE to execute the simulation protocol and to access the workflow conveniently [19].

In this work we combine the workflow engine Galaxy with Singularity to increase the reproducibility and mobility of scientific workflows. In the first part we describe how we achieve a containerization of the workflow. Then we show that the containerized workflow is straightforward executable on a local machine, on a HPC-system, and in the cloud. Last we show how our approach differs from related work that tries to make scientific workflows more reproducible.

II. Containerization of Galaxy workflows

The containerization is a fully automated process in two distinct steps as shown in Figure 1. The first step creates a template Singularity container with an operating system and installs a Galaxy instance in it. In the second step the actual workflow and its tool dependencies are installed in the container. Both encapsulation steps are explained in more detail in this section. The whole code base, configuration files and the example workflow are accessible via a GitHub repository [20]. We used Singularity 2.3.2, Python 2.7.13 and the Python package bioblend 0.9.0 for this work.

A. Create a template container

In this step a template Singularity container is created that is able to store one or more Galaxy workflows. We provide a shell-script that creates the container by bootstrapping with a so-called Singularity definition file that describes how a Singularity container is provisioned.

During the bootstrap we install CentOS 7.3 in the container, but also other operating systems would be possible. Subsequently several system packages are installed by the yum pack-

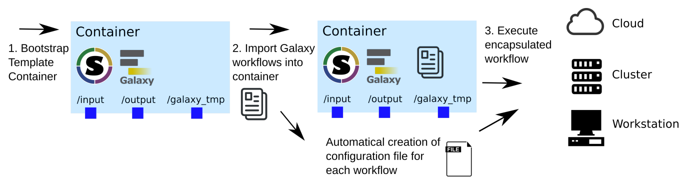
1. Bootstrap Template Container
Container
Galaxy
/input
/output
/galaxy_tmp
2. Import Galaxy workflows into container
Container
Galaxy
/input
/output
/galaxy_tmp
Automatical creation of configuration file for each workflow
3. Execute encapsulated workflow
Cloud
Cluster
Workstation

Fig. 1. Overview of the containerization workflow. **1.** Create the template Singularity container with an operating system and Galaxy. Also three folders are created to serve as mount points for the input, output and temporary Galaxy directory. **2.** Import the Galaxy workflows. For each workflow a configuration file is created **3.** Specify input/output mounts and workflow parameters using the execution configuration file. Then run the workflows in the cloud, on a cluster, or your own workstation.

age manager: `Development-Tools, epel-release, python-pip, python-devel, bzip2, git.`

Finally Galaxy is installed by cloning the GitHub branch corresponding to the user specified Galaxy version. We start Galaxy one time to initialize the Galaxy instance. After the initialization completes, we shut down the initialized Galaxy and the template container is ready to encapsulate Galaxy workflows.

B. Encapsulate Galaxy workflows in template container

The template container from above is now used to encapsulate Galaxy workflows. Because Singularity containers behave like normal files on the file system one can copy the template container and use the copy to encapsulate other workflows.

Galaxy workflows are explicitly described in .ga files that can be exported from other Galaxy instances or downloaded from platforms like myExperiment. Our import script will automatically install the workflow as described in the workflow file. It is also possible to obtain the workflow from a running Galaxy instance, but you need access to the workflow in the Galaxy instance beforehand.

We wrote a Python script to import the workflow in the container automatically. Prior to the workflow import, certain settings are specified in a configuration file. During the import, we generate another configuration file that is needed to execute the encapsulated workflow afterwards. First, we startup Galaxy in the container and use the Python module bioblend [21] to interact with the Galaxy-API.

a) Create new Galaxy user: We need a Galaxy user in the container in order to run the workflow and to use important Galaxy features like data libraries. Therefore a new Galaxy user is created in the containerized Galaxy instance. One has to specifiy beforehand the user name and user mail-address in the configuration file. We also generate an API-key as well as a random password for this user and store them in the execution configuration file. The API-key is used to run the containerized workflow from the command line whereas the password is needed to access the Galaxy instance in the browser.

b) Install tools and import workflow: Galaxy describes a workflow explicitly in a .ga-file that contains a json-dictionary. Among others, the file lists the inputs and computational steps of the workflow. For the input steps we parse the required input datasets together with their Galaxy datatype and an additional description if provided. These input information are written to the execution configuration file. Later the user specifies the files that should serve as input when the workflow is executed on the command line.

For the computational steps we parse the specified tool version and install everything by using the Galaxy-API. The tools and their dependencies are then installed from the publicly available Galaxy ToolShed. Also so-called runtime parameters are handled explicitly. Runtime parameters are parameters whose values are not specified in the workflow file. The user has to specify them when invoking the workflow, e.g. number of iterations or cutoff values. The runtime parameters for a tool are also written to the execution configuration file.

III. Execution of the Encapsulated Workflow

Additionaly to Singularity the Python code presented in this paper has to be installed on the remote system to run the containerized Galaxy instance. Whereas Singularity has to be installed by an administrator, the Python code can reside in user space as it does not need any special privileges. The user has to upload the container to the system that will execute the container.

We provide a Python script that can execute the containerized workflow in two modes, an automatic and an interactive mode. To execute the workflow automatically the user specifies the path to the input files and parameters of the computational steps in the execution configuration file. This file was automatically created in the encapsulation step and needs just small adjustments. The workflow is started by executing:

```
python execute_workflow.py --conf <ini-file>
```

The execution script starts Galaxy, uploads the input files, invokes the workflow, waits until the workflow has completed, and downloads the workflow results to a specified output folder. The input files are not duplicated when they are uploaded to Galaxy but a symbolic link to the location of each input file on the file system is created in the Galaxy input data library.

In the interactive mode, Galaxy is started by the script and the user can then access the Galaxy instance via the browser. The user can login to Galaxy, study the containerized workflows, upload datasets to the Galaxy instance, and run the workflow as well as just single tools. In the end the user can also download the resulting datasets.

Because Singularity container cannot be changed by a non-root user, we had to change the default Galaxy configuration in some points for the execution. Most of the actions in Galaxy cause changes of the Galaxy database, writes to logging files, or leads to the creation of some temporary data. Therefore we created a temporary directory for Galaxy in the container where the database and temporary files reside. Upon execution we create a unique temporary directory on the host and mount it to Galaxy's temporary directory in the container. All write operations of Galaxy are performed in this temporary directory on the host. This also ensures that the container itself is not altered during the execution. The temporary directory on the host is deleted after the workflow was executed.

IV. Containerization increases reproducibility

The created Singularity container encapsulates immediately executable workflows. This fact alone increases the reproducibility and reuse massively because it eliminates some of the most important reasons of workflow decay like missing execution environment or an insufficient description of the workflow [15].

Sharing the container together with the execution configuration files enables other researchers to reproduce published results or adjust parameters of the workflows to run their own analysis. Archiving the container ensures in principle reproducibility and reuse of the workflow a long time after creation. The CiTAR project [1] for example has the aim to archive Docker and Singularity containers and provide an infrastructure that secures the ability to execute the container in the future.

The proposed method increases also the portability of the computation. One can develop the workflow in a local or public Galaxy instance, encapsulate the workflow and run it on any computational resource. We did this for a 14-step Next Generation Sequencing workflow that is also included in the GitHub repository of this project [20]. We encapsulated the workflow in a Singularity container and were able to run the workflow on BinAC [22] and a VM on the de.NBI cloud site Tübingen[2].

V. Related Work

Research environments and workflows implemented in Galaxy are often shared via virtual machine images [23], [24]. These virtual machine images contain all dependencies and are ready-to-use. But virtual machine images are usually not executable on HPC clusters because just few clusters installed hypervisors for the virtual machines on the compute nodes.

Galaxy is available as a Docker container [25] and the Galaxy team also provides the library Ephemeris to interact with the Galaxy-API via bioblend to install workflows and its dependencies[3]. Both, the Docker Galaxy container and Ephemeris, could be combined to encapsulate scientific workflows in a Docker container. This would be a subset of the functionality our approach provides. Besides we are using a different containerization technique, we also provide automatic execution of the workflow and handle input and output data. Singularity containers are more suitable for scientific HPC clusters because Singularity targets scientific applications and supports HPC resources like MPI (Message Passing Interface), Infiniband, and GPUs [17]. Another aspect is that Docker needs a root owned daemon process whereas Singularity tries to minimize the security risk with suid-executables that handle the privilege escalations needed for software containerization.

[1]https://www.alwr-bw.de/kooperationen/bwzwm/
[2]https://denbi.uni-tuebingen.de
[3]https://github.com/galaxyproject/ephemeris

The project ViCE [26] works on sustainable virtual research environments like Galaxy. ViCE offers a image registry for virtual collaborative environments and an infrastructure to support accessibility and reproducibility of the environments.

VI. Outlook

Singularity will be further developed. At the time of writing Singularity 2.4 was released. The new version offers some new features of which the containerization of workflows benefits, e.g. new container format, support of overlays and container instances.

Also Singularity could be used to encapsulate other workflow engines like Taverna or Knime. The created containers could be shared via newly platforms like Singularity Container Registry [4].

[4]https://singularity-hub.org/

References

[1] D. Waltemath, R. Adams, D. A. Beard, F. T. Bergmann, U. S. Bhalla, R. Britten, V. Chelliah, M. T. Cooling, J. Cooper, E. J. Crampin, A. Garny, S. Hoops, M. Hucka, P. Hunter, E. Klipp, C. Laibe, A. K. Miller, I. Moraru, D. Nickerson, P. Nielsen, M. Nikolski, S. Sahle, H. M. Sauro, H. Schmidt, J. L. Snoep, D. Tolle, O. Wolkenhauer, N. Novère, "Minimum information about a simulation experiment (MIASE)", *PLoS Computational Biology*, vol. 7, no. 4, pp. 1–4, 2011

[2] A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, M. Vingron, "Minimum information about a microarray experiment (MIAME)-toward standards for microarray data", *Nature Genetics*, vol. 29, no. 2, pp. 365–371, 2001

[3] G. K. Sandve, A. Nekrutenko, J. Taylor, E. Hovig,"Ten Simple Rules for Reproducible Computational Research", *PLoS Computational Biology*, vol. 9, no. 10, pp. 1–4, 2013

[4] J. P. A. Ioannidis, D. B. Allison, C. A. Ball, I. Coulibaly, X. Cui, A. C. Culhane, M. Falchi, C. Furlanello, L. Game, G. Jurman, J. Mangion, T. Mehta, M. Nitzberg, G. P. Page, E. Petretto, and V. van Noort, "Repeatability of published microarray gene expression analyses", *Nature Genetics*, vol. 41, no. 2, pp. 149–155, 2008.

[5] K. A. Baggerly and K. R. Coombes, "Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology", *Annals of Applied Statistics*, vol. 3, no. 4, pp. 1309–1334, 2009.

[6] D. Garijo, S. Kinnings, L. Xie, L. Xie, Y. Zhang, P. E. Bourne, and Y. Gil, "Quantifying reproducibility in computational biology: The case of the tuberculosis drugome", *PLoS ONE*, vol. 8, no. 11, pp. 1–11, 2013.

[7] J. Goecks, A. Nekrutenko, J. Taylor, and T. G. Team, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences", *Genome Biol*, vol. 11, no. 8, pp. 1–13, 2010.

[8] D. Blankenberg, G. V. Kuster, N. Coraor, G. Ananda, R. Lazarus, M. Mangan, A. Nekrutenko, and J. Taylor, "Galaxy: A web-based genome analysis tool for experimentalists", *Current protocols in molecular biology*, pp. 1–21, 2010.

[9] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor, W. C. Miller, W. J. Kent, and A. Nekrutenko, "Galaxy: a platform for interactive large-scale genome analysis", *Genome research*, vol. 15, no. 10, pp. 1451–1455, 2005.

[10] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A. Nieva de la Hidalga, M. P. Balcazar Vargas, S. Sufi, and C. Goble, "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud.", *Nucleic acids research*, vol. 41, no. Web Server issue, pp. 557–561, 2013.

[11] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel, "KNIME: The Konstanz Information Miner", in *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*, Springer, 2007.

[12] J. Krüger, R. Grunzke, and S. Gesing, "The MoSGrid Science Gateway–A Complete Solution for Molecular Simulations", *Journal of Chemical Theory and Computation*, vol. 10, no. 6, pp. 2232–2245, 2014.

[13] L. Zimmermann, R. Grunzke, and J. Krüger, "Maintaining a Science Gateway – Lessons Learned from MoSGrid", *Proceedings of the 50th Hawaii International Conference on System Sciences*, pp. 6233–6242, 2017.

[14] D. D. Roure, C. Goble, and R. Stevens, "The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflow", *Future Generation Computer Systems*, no. November 2007, pp. 561–567, 2009.

[15] J. Zhao, J. M. Gomez-Perez, K. Belhajjame, G. Klyne, E. Garcia-Cuesta, A. Garrido, K. Hettne, M. Roos, D. De Roure, and C. Goble, "Why Workflows Break -Understanding and Combating Decay in Taverna Workflows", (Washington, DC, USA), pp. 1–9, IEEE Computer Society, 2012.

[16] "Docker, https://www.docker.com/."

[17] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute", *PLOS ONE*, vol. 12, no. 5, pp. 1–20, 2017.

[18] C. Boettiger, "An introduction to Docker for reproducible research", *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, 2015.

[19] M. Hanussek, F. Bartusch, J. Krüger, and O. Kohlbacher, "Efficent Mass Spectra Prediction through Container Orchestration with a Scientific Workflow", in *9th International Workshop on Science Gateways (IWSG)*, 2017.

[20] F. Bartusch, "fbartusch/galaxy2singularity: v.0.1.0, 10.5281/zenodo.1122568."

[21] C. Sloggett, N. Goonasekera, and E. Afgan, "BioBlend: Automating pipeline analyses within Galaxy and CloudMan", *Bioinformatics*, vol. 29, no. 13, pp. 1685–1686, 2013.

[22] J. Krüger, V. Lutz, F. Bartusch, W. Dilling, A. Gorska, C. Schäfer, and T. Walter, "Bioinformatics and astrophysics cluster (BinAC)", *3rd bwHPC Symposium*, pp. 1–1, 2017.

[23] S. J. Schultheiss, G. Jean, J. Behr, P. Drewe, N. Görnitz, A. Kahles, P. Mudrakarta, V. T. Sreedharan, G. Zeller, and G. Rätsch, "Oqtans: a Galaxy-integrated workflow for quantitative transcriptome analysis from NGS Data", *BMC Bioinformatics*, vol. 12, no. Suppl 11, pp. 1–2, 2011.

[24] R. L. Davidson, R. J. M. Weber, H. Liu, A. Sharma-Oates, and M. R. Viant, "Galaxy-M: a Galaxy workflow for processing and analyzing direct infusion and liquid chromatography mass spectrometry-based metabolomics data", *GigaScience*, vol. 5, no. 1, pp. 1–9, 2016.

[25] B. Grüning, M. van den Beek, B. Batut, J. Chilton, M. ISHII, D. Ryan, E. Afgan, H. Rudolph, K. Ellrott, C. Smith, P. Moreno, H.-R. Hotz, G. V. Kuster, R. Baertsch, M. Edwards, G. L. Corguillé, A. Azab, S. Hiltemann, Rdmorin, M. Chambers, T. Tanjo, R. Hernández, Jasper, and A. Petkau, "bgruening/docker-galaxy-stable: Galaxy Docker Image 17.05, 10.5281/zenodo.583723."

[26] C. B. Hauser, J. Domaschka, "ViCE Registry: An Image Registry for Virtual Collaborative Environments", *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 82–89, 2017