EPiC Series in Computing

Volume 107, 2025, Pages 165–174

Proceedings of EUNIS 2025 annual congress in Belfast

Rapid Development Moving between Public Cloud and On-Premises

Marius Politze[1*], Bernd Decker[1†] and Uta Christoph[1]

[1] RWTH Aachen University, Germany

politze@itc.rwth-aachen.de, decker@itc.rwth-aachen.de,
christoph@itc.rwth-aachen.de

Abstract

Within this paper we present our approach towards using public cloud services within for speeding up development and deployment times for university-wide software services. Within three use cases we discuss challenges faced by development teams in adapting workflows. By employing Infrastructure as Code principles, teams can standardize environment setups and automate provisioning processes, significantly reduce setup times, and allow migration to on-premises infrastructures for sustainable operation. This demonstrates the value of being able to strategically choose between public cloud and on-premises services to accelerate software development cycles.

1 Introduction

Digitization is a constant driver in higher education and research. Universities' IT services, therefore, are constantly professionalizing their workflows. Hence, within the IT Center of RWTH Aachen University, several departments are performing genuine software development to support members of the universities within their various roles: students, teachers, researchers and administration. While the extent and contextual bounding conditions vary greatly based on the supported processes, the overall environment as well as technical bounding conditions are often comparable or even identical.

As part of this case study, we present three use cases from two departments targeting different audiences within the university: "university-wide, student life-cycle management and e-learning services", "research data management services" and "lecture support". All use cases were carried out mostly independently from one another within different teams, with their own agenda and time management. However, involved lead engineers and architects were given the freedom to peek into the other projects and thus could directly share experiences.

* https://orcid.org/0000-0003-3175-0659

† https://orcid.org/0000-0002-9627-5695

From the perspective of application, the teams are facing quite different challenges, but due to the organizational structure of the IT Center, they technically have several commonalities: Traditionally, departments that are devoted towards software, process development and consulting do not operate their own hardware but internally rely on server infrastructures operated in a model that would be considered a private cloud environment today. While this reduces control over the execution environment this enforces a strong separation between service development and service operations. Hence, we often see a clear separation of concerns in different teams for operation of IT systems, development of custom solutions, and adaptation of existing, often commercial, products.

Overall, this makes it especially hard to extensively adopt invasive technologies like Kubernetes that would require a synchronized effort crossing all involved teams at the same time. We see this also reflected in other university ICT providers, generally adopting cloud technologies rather slowly (Okai, Uddin, Arshad, Alsaqour, & Shah, 2014; Gholami, Daneshgar, Beydoun, & Rabhi, 2017). Furthermore, long term experiences and impact as well as mobility or dependency to certain vendors remains partially unclear (Kratzke & Quint, 2017).

In this paper, we present a combined list of experiences and best practices from several teams totaling about 35 developers and stakeholders ranging from students, university staff, and faculty members to externals in third-party-funded projects. Based on these experiences this paper aims to provide a consolidated answer to the following research questions:

- RQ1: Under which conditions can public cloud offers complement on-premises IT operations?
- RQ2: What factors need to be considered to ensure mobility of application between cloud providers or into the private cloud.

2 Current Situation

Different teams have adopted a wide range of cloud technologies for different phases of their software-life-cycle and to support their daily work. Based on the different teams' shared experiences, the following sections give an overview of these in the form of best practices implemented in conjunction with public cloud providers, where they support software development and adjacent processes. To support their engineering process, the teams use SCRUM based workflows that are technically supported by GitLab (Politze & Christoph, 2020; Politze, et al., 2023). This gives a state-of-the-art environment for professional software development and forms a common starting point for the software development teams.

2.1 Private Cloud Development Environments

The actual implementation work is done on personally assigned virtual machines resembling the production environments as close as possible or rarely on local computers / laptops. Development and production environment synergically share the same infrastructure, making the mode of operation economically efficient. Hence, IT operations personnel is responsible for maintaining the development environments. While the close connection of development and production environments greatly helps ensuring transferability and reproducibility of issues, it drastically increases reaction times to changing hardware requirements. This is mostly due to communication overhead between the operations and development departments. Changes on development environments are competing against maintenance work for the production systems. Consequently, this reduces the innovative potential especially when introducing disruptive technologies like containers or Kubernetes that would require significant deviation of infrastructures.

Issues with this mode of operation are further pointed out as more funding is coming from third parties and therefore connected with more binding timelines compared to basic funding. While this has increased innovation potentials, it put even more pressure on advancing IT infrastructures – first for development of new services and enhancement of existing services and finally for putting these services into operation. To ensure success of these projects it was therefore necessary to reduce time-to-market and to reduce friction within development, prototyping and finally deployment of these services.

2.2 Advancing Development Infrastructures

An initially proposed solution to the problem was to separate the development and production infrastructures and make them more or entirely independent. While this would provide complete freedom for developers, it would also come with the burden of having to maintain a so far entirely unknown technology stack, including server hardware, virtualization stacks and operating systems that were previously not within the scope of the development team. Apart from additional workload for the development team, setting up an entirely new system also jeopardizes the economic synergies of the current joint operating model. Consequently, this approach was quickly disregarded.

To reduce direct infrastructure costs and limit the depth of technical knowledge acquisition, the groups decided to make use of public cloud offers that allow scaling infrastructures as needed and provide ready-to-use virtualization environments that require little to no additional knowledge. At the same time these environments allow for rapid prototyping as they feature a set of clearly defined interfaces that can be directly used and integrated into running projects and allow gradual adaption within the development teams. Furthermore, individual teams could choose the right technology partner for their current project. This was especially easy since many public cloud providers have signed the OCRE contracts with Géant which in turn allowed quick and low-threshold subcontracting for small projects as well as discounted rates.

3 Use Cases

As described previously several project teams made the transition towards the new mode of operation in parallel. This allows to analyze the slightly independent approaches taken and allow a slightly broader evaluation. All presented use cases have the goal of creating a rapid development model and to one extend or another cover the range from development of a new software to deployment of a prototype for a designated group of users.

3.1 Development Environment: Coscine.nrw

For development environments, the use case illustrates the provisioning of development sandboxes for teams with approximately 30 individuals in total contributing to the open source data management platform Coscine.nrw and DataStorage.nrw (Politze, et al., 2020; Politze, Heinrichs, Hunke, Lang, & Eifert, 2025). The team members have diverse roles and expertise ranging from full stack software development to high level quality assurance and service management. The goal of this use case was to allow setup of defined development environments for the teams, especially the creation of multiple virtual machines per person, with limited machine lifespans, typically one to twelve months. Moving these environments into the public cloud additionally has the advantage of better isolation for the production environments.

Based on experiences gained in the FAIR Data Spaces project, the team decided to make use of the Open Telekom Cloud (OTC), a Germany based "super scaler" with data centers in Germany and The Netherlands (Politze, 2022). A thing that was noted while working on the demonstrator was that the discounted rates were only valid for the Dutch datacenter of OTC. While this had little to no technical effect, it shows that even for presumably simple projects cost models of providers need to be analyzed and a cost controlling needs to be established to avoid overspending.

Technically, the use case is set up using Terraform. Terraform provides a configuration language and a toolset to define (cloud) infrastructure as code (IaC). Hence, the description of the development infrastructure is standardized through predefined templates and managed via a shared Git repository, ensuring consistency and facilitating automation through GitLab CI pipelines. This approach greatly suited the development team that was used to work with GitLab and source code in general and allowed a more natural handling of infrastructure topics and reduced the necessary depth of knowledge acquisition needed for the overall team. The integration of Terraform with GitLab CI furthermore enables automated infrastructure provisioning and management. Terraform scripts define the desired state of cloud resources, which are executed through GitLab CI pipelines upon code commits. This approach ensures that infrastructure changes are version-controlled, reviewed, and consistently deployed, enhancing reliability and reducing the likelihood of configuration drift. Terraform modules encapsulate reusable infrastructure components, promoting modularity and reducing complexity in infrastructure definitions. This modular approach facilitates scalability and maintainability, essential for managing large and dynamic cloud environments.

Additionally, to IaC code, the setup required an explicit definition on how to set up development environments. This was realized with a set of scripts that are executed on provisioning of the virtual machines. Making these setup steps explicit was putting additional workload on the team but in turn had a quick payoff in documenting and unifying setup of machines – for development and ultimately for production environments. These definitions allowed the development team on the one hand to adopt container-based application deployment and on the other hand helped IT operations personnel transferring the requirements to the local infrastructure.

As shown in Figure 1 the development network consists of a set of development servers – typically one for each member of the development team. Connection from the workstations are tunneled through proxy servers for SSH and HTTPs. This was done to cut costs for external IP addresses; development servers are only connected to an internal network. For the different types of servers specialized provisioning scripts ensure reproducibility of the setup.

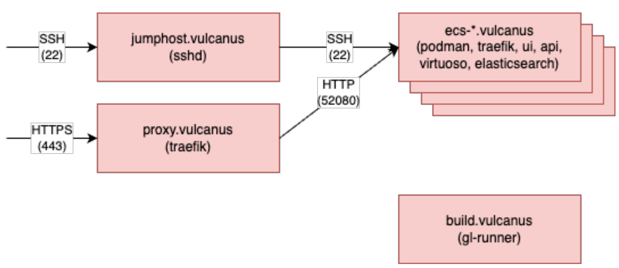

Figure 1: Schematic overview of the created infrastructure within the development environment.

The resulting development infrastructure closely resembles the structures that were maintained on-premises previously. Hence, only a fraction of the cloud native features offered by the OTC were put into practice. In turn this allows synchronization of cloud and on-premises infrastructures that is needed to put services into long term operation.

3.2 LLM Development: KI:connect.nrw

In the second use case, the project KI:connect.nrw is presented. The goal of this project is to provide data protection-compliant access to generative Artificial Intelligence (AI) for employees and students of universities within the German federal state North Rhine-Westphalia (NRW). Since the introduction of ChatGPT in November 2022 and the subsequent discussions regarding the impact of AI on research, teaching, and administration, the demand for powerful AI solutions has increased significantly. Accordingly, there was considerable pressure to provide an adequate solution promptly. A key prerequisite was that the developers could quickly familiarize themselves with the new technologies while concurrently establishing the necessary infrastructure. The critical components included, in particular, Large Language Models (LLMs) and vector databases for Retrieval-Augmented Generation (RAG).

To meet these requirements the team decided to host the development infrastructure in the Microsoft Azure Cloud. In coordination with the IT operations team, care was taken to select only technologies that could also be hosted on-premises at a later stage. This strategic choice ensured that the project's advancements would be sustainable and transferable to local infrastructure, aligning with long-term operational goals.

Since existing contracts for the use of Azure were already in place, development could commence immediately. By leveraging the cloud infrastructure, the team significantly reduced the setup time typically required for physical infrastructure, thereby accelerating the project's time to market. Via Azure OpenAI, endpoints to different LLMs were provided. Additionally, the necessary open-source infrastructure for RAG was constructed using a Qdrant vector database, Fileservers and NoSQL database for logging and billing processes (see Figure 2 for the development system architecture). In parallel, the IT operations team began acquainting themselves with the RAG infrastructure and implement it on-premises, while a partner project started with the local hosting of open-source LLMs. This strategy allowed different work packages, which would typically be handled sequentially, to be processed simultaneously.

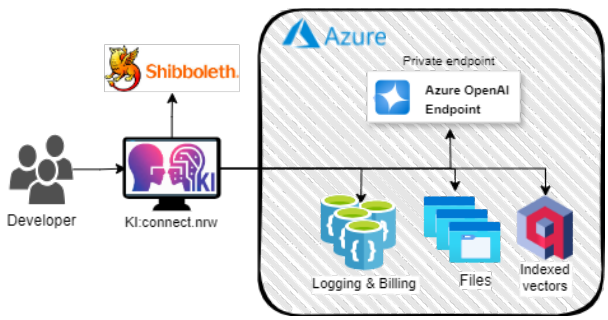

Figure 2: KI:connect.nrw development system overview.

For the utilization of the LLMs, the "pay-as-you-go" cost model was chosen, whereby only the actually consumed prompt tokens are billed. In the development phase, token consumption was minimal, as only the development team accessed the endpoints. Furthermore, small and cost-effective LLMs were used, given that the focus was on the technical implementation rather than the quality of the generated responses. This strategy minimized the monthly costs for this aspect of the development. The sizing of the RAG infrastructure was also deliberately kept modest, resulting in low expenses. Consequently, the monthly costs for the development infrastructure were maintained well below €100.

This approach allowed the KI:connect.nrw project to rapidly develop a compliant and scalable AI solution, effectively meeting the urgent demands of the academic community while laying a solid foundation for future on-premises deployment (see Figure 3 for the current system architecture).

3.3 Lecture Environment

A use case slightly outside of the core development team is the support of the lecture “Large Scale IT and Cloud Computing”. The lecture aims to teach students basics of scalable IT systems. Scalability is regarded from several viewpoints: business process, software architecture, implementation and operations. While most concepts can be taught on a purely theoretical basis on the students’ laptops, we decided to take a more practical approach: Applying the principles of participatory live coding (Nederbragt, Harris, Hill, & Wilson, 2020) the students should implement and deploy a simple IT supported business process to cloud infrastructures. The requirements in this scenario include high

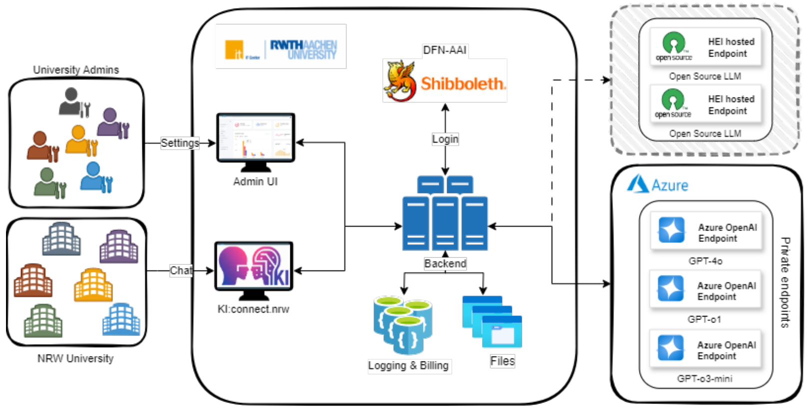

Figure 3: KI:connect.nrw overall system architecture.

autonomy for the students, short machine lifespans (ranging from one to fourteen days), and the ability to scale for large participant numbers.

In an initial iteration of the lecture, we used local infrastructures of the IT Center. In this case we had potentially hundreds of students operating virtual machines within the universities' IT infrastructure. This setting had several drawbacks: a) allowing students to operate within the infrastructure potentially exposes high risks and b) the one-time setup overhead is high and again involves teachers responsible for the lectures as well as IT operations personnel. Students and their results had to be monitored intensively effectively reducing their autonomy for trying out the presented technology. Also, they are limited by the explicitly offered setups.

This previous approach proved to be hardly scalable and switching to a public cloud-based model was desirable to address exiting drawbacks. Our initial iteration leveraged Microsoft Azure through education grants for approximately 20 participants, expanding to Google Cloud for around 120 participants. The existing education grants allowed each student to individually create an account and have complete freedom to experiment with the infrastructure. On the other hand, it required the students to track costs themselves as the education grants are limited to currently approximately 100€ and students must retain enough funds to complete all contents covered within the lecture over the progress of the semester. Certainly, this makes the lecture dependent on the cloud providers generosity to provide education grants. Transferring the educational grants to the OCRE offers is very well possible but requires substantial funding for the lecture and a proper identity management if potentially hundreds of students need to be authorized to access the cloud environment. Quota limits help reducing the risk of overspending.

The creation and deployment of virtual machines, as well as the installation of container runtimes were taught as part of the lecture, but students were not limited to that. The cloud setup greatly increased autonomy of the students being able to use the entire available offers of the respective cloud, it makes reproducibility of their results harder for teachers. Consequently, students were obliged to document the necessary infrastructures as code using Terraform. Terraform automates the deployment and teardown of their environments, ensuring efficient resource utilization and management. This approach additionally allows students to save on their limited funds by making use of the teardown feature and thus only paying for infrastructure while it is used during the lecture.

Overall, the integration of real world cloud infrastructure provides relevant and realistic experiences for the students and provides insights on infrastructure management as well as the resulting direct costs of infrastructures.

4 Evaluation & Lessons Learned

In the previous section the three use cases presented the approach of different teams towards outsourcing workload into the public cloud. All scenarios were carried out intending to allow migration of already operated infrastructure from the public cloud offers to on-premises infrastructures.

Within the development environment use case the team was able to reduce the setup time of a development server from approximately two weeks to roughly 5 minutes. The majority of time savings comes from the reduction of manual steps and idle times after handovers between development and IT operations team within the provisioning process. Similarly, in the KI:connect.nrw use case, leveraging cloud-based services allowed the development team to circumvent the delays typically associated with setting up on-premises infrastructure, thereby achieving a significantly faster time to market. The immediate availability of Azure resources enabled the team to rapidly initiate the development process and gain hands-on experience with LLMs and RAG technologies.

From both the lecture case and the development environment case, we see the IaC definitions are key for reproducibility and fast setup times. Especially for costly infrastructure components separation in stateful and stateless parts is essential to plan teardown effectively and without the risk of data loss. While the mostly experimental scenario of the lecture has little to no risk, development environments may contain uncommitted code and work of several days. Clear guidelines for committing code to GitLab are required to minimize the risk of loss. Changes on individual machines need to be documented within the IaC to assure reproducibility of infrastructure setup after a potential teardown. This was a significant change in the way developers worked e.g. needing to document package installations in startup scripts or container images once they have stabilized.

Cost management was a critical aspect across all use cases. While cloud providers document these costs rigorously and send bills accordingly it is more a matter of understanding their models. In the KI:connect.nrw project, the use of the "pay-as-you-go" model and the deployment of small, cost-effective LLMs kept operational expenses low during the initial development phase. This financial prudence allowed the project to remain within budget constraints while still achieving its technical objectives promptly. For on-premises we see that many IT centers within the HEI context do not, and sometimes even cannot, fully assess costs of their infrastructures. This complicates truly comparative analysis of costs.

As part of this study, we conducted a comparative cost analysis between on-premises virtual server hosting and leveraging the Open Telekom Cloud. This revealed significant differences in infrastructure and personnel costs. On-premises solutions incur higher initial costs from setup, software deployment, and configuration. In contrast, the public cloud offers scalable pricing models with lower upfront costs and reduced personnel overhead, especially scaling well for frequent setups and teardowns. Looking at costs, both for on-premises as well as cloud infrastructures requires insights into cost models.

Supporting these different use cases required being able to access public cloud services in short time frames. The OCRE contract framework provided by Géant posed a more than sufficient starting point: Most importantly the agreements with cloud providers can be arranged ahead of time – especially before first projects express explicit demands. This can greatly speed up response time to provision the tenants required for individual use cases. A widespread integration into provisioning, accounting, and billing processes, e.g. to separate bills by organizational unit or even project, can be explicitly demanding for the university administration and for the cloud provider. If an organization plans to make use of hyper scalers in the future it is worth investigating these issues before first use cases are technically starting off as contracting and financial issues may inhibit forthcoming in later phases of the projects.

Even though the three use cases come from different parts of the service spectrum and mostly independent teams, retrospectively we can observe some common choices: All teams decided to restrict their usage of the offered features mostly to Infrastructure as a Service (IaaS) offers or to APIs that are specified on an abstract and functional level. Effectively the teams set up virtual machines and respective network infrastructures and then deployed workloads mostly traditionally running containerized applications. While all considered cloud providers had more elaborated “cloud native” Platform Services, e.g. “Kubernetes as a Service”, “Database as a Service” or “Functions as a Service” readily available this was a deliberate choice to ensure an exit scenario where the cloud infrastructure would need to be transferred to the on-premises environment. Clearly this puts additional workload on the teams setting up the infrastructure but on the other hand this approach, evident in the KI:connect.nrw project, ensured that the project's advancements would be sustainable in the long term, facilitating a smoother transition from cloud-based to local infrastructure.and ensures digital sovereignty.

Finally, we want to reconsider the initially posed research questions:

- RQ1: Under which conditions can public cloud offers complement on-premises IT operations?

We have shown in the three use cases that public cloud offers, both from hyper scalers as well as from European “super scalers” can complement IT infrastructures readily available on-premises. This is especially due to shorter provisioning cycles. In phases of the use cases where environments are subject to frequent changes or very specialized hardware was required, the projects could profit from the cloud model. In turn local operations profited from more elaborated and well documented requirements after initial development and prototyping phases.

- RQ2: What factors need to be considered to ensure mobility of application between cloud providers or into the private cloud.

A great selling argument by cloud providers are integrated Platforms as a Service (PaaS). While these services offer mostly simple interfaces to hide complex infrastructures like a database service or a deployed LLM, usage of these interfaces needs to be well considered as they likely increase dependency to the interfaces offered by the cloud provider. This reduces mobility between different cloud offers but also complicates exit scenarios. Within the presented use cases we have shown that by restricting to IaaS service the teams could greatly reduce their dependencies. PaaS interfaces were evaluated thoroughly and verified to be replicable using (locally) deployed software. Even after years of advancement of cloud technologies interoperability between cloud offer still is a major issue for mobility (Kratzke & Quint, 2017).

5 Conclusion

Within the three presented use cases we have shown different approaches on outsourcing parts of the development and production environments into public cloud offers. Due to new changed funding requirements, increased pressure to modernize software development and service deployment environments or changing demands for hardware. We clearly see the public cloud offers to supplement available local infrastructures rather than replacing them.

Once requirements are known and have stabilized, running services on-premises allows using synergies during operation and retains digital sovereignty. This effectively reduces workload on IT operations teams and increases time-to-market speed of development teams without the need of permanently operating parallel infrastructures. The definition of IaC makes setup of the environments more reproducible and served as a common means of communication between development teams and IT operations.

We greatly profited from the OCRE contracts allowing us to flexibly use cloud offers without tender processes and streamlining procurement. This was especially true for Microsoft Azure with whom our university had a contract even before the describes use cases started. Existing Azure contracts enabled immediate development initiation, contributing significantly to the reduced time to market. We can clearly advise to strategically pick some partners and start the contracting process even before concrete use cases are established. In our environment we made sure to have at least one hyper scaler and one European provider available to ensure digital sovereignty. While cloud providers allow virtually infinite scalability we can easily profit for ad-hoc and high peak scenarios, but teardown processes and cost models need to be understood thoroughly for larger deployments to prevent surprisingly high costs.

Additionally, we conclude to ensure the exit scenario while setting up public cloud use cases. In the described use cases we mostly refrained from using software as a service (SaaS) offers but built the services using infrastructure as a service (IaaS) components and available open source software. In the case of SaaS LLMs we ensure that local deployments comply to the same interfaces. This ensured portability of the provisioned infrastructure between cloud providers and on-premises.

The projects demonstrate how strategic use of cloud services can accelerate development processes, reduce time to market, and lay the groundwork for future on-premises operations. The methodologies serve as valuable models for similar initiatives aiming to balance rapid deployment, cost efficiency, and long-term sustainability in technology adoption within the higher education sector.

6 Acknowledgements

DataStorage.nrw is funded by Ministerium für Kultur und Wissenschaft des Landes Nordrhein-Westfalen (MKW: 214-76.01.09-7-7937 DFG: INST 222/1530-1), KI:connect.nrw and Coscine.nrw are funded by Ministerium für Kultur und Wissenschaft des Landes Nordrhein-Westfalen as part of the statewide digitalization strategy.

The conceptual work was supported with resources granted by NFDI4Ing, funded by Deutsche Forschungsgemeinschaft (DFG) under project number 442146713, NFDI-MatWerk, funded by Deutsche Forschungsgemeinschaft (DFG) under project number 460247524.

7 References

Gholami, M. F., Daneshgar, F., Beydoun, G., & Rabhi, F. (2017). Challenges in migrating legacy software systems to the cloud — an empirical study. *Information Systems, 67*, 100–113. doi:10.1016/j.is.2017.03.008

Kratzke, N., & Quint, P.-C. (2017). Understanding cloud-native applications after 10 years of cloud computing – A systematic mapping study. *Journal of Systems and Software, 126*, 1–16. doi:10.1016/j.jss.2017.01.001

Nederbragt, A., Harris, R. M., Hill, A. P., & Wilson, G. (2020). Ten quick tips for teaching with participatory live coding. *PLoS computational biology, 16*. doi:10.1371/journal.pcbi.1008090

Okai, S., Uddin, M., Arshad, A., Alsaqour, R., & Shah, A. (2014). Cloud Computing Adoption Model for Universities to Increase ICT Proficiency. *SAGE Open, 4*. doi:10.1177/2158244014546461

Politze, M. (2022). Hybrid Cloud Scaleout: Orchestrating Workloads with GitLab. In J.-F. Desnos, R. Yahyapour, & R. Vogl (Ed.), *EPiC Series in Computing: Proceedings of EUNIS 2022 – The 28th International Congress of European University Information Systems. 86.* EasyChair. doi:10.29007/nwh7

Politze, M., & Christoph, U. (2020). Migrating from Team Foundation Server to GitLab – A Progress Report. In *Proceedings of the EUNIS 2020 Congress* (pp. 51–53). Helsinki, Finland. Retrieved from https://www.eunis.org/download/2020/EUNIS_Book-of-Abstract_2020.pdf

Politze, M., Christoph, U., Decker, B., Hristov, P., Lang, I., Nelesen, M., & Yazdi, M. A. (2023). Supporting Software Development Processes for Academia with GitLab. *EPiC Series in Computing: Proceedings of European University Information Systems Congress 2023.* EasyChair. doi:10.29007/9157

Politze, M., Claus, F., Brenger, B., Yazdi, M. A., Heinrichs, B., & Schwarz, A. (2020). How to Manage IT Resources in Research Projects? Towards a Collaborative Scientific Integration

Environment. In *Proceedings of the EUNIS 2020 Congress* (pp. 45–48). Helsinki, Finland. Retrieved from https://www.eunis.org/download/2020/EUNIS_Book-of-Abstract_2020.pdf

Politze, M., Heinrichs, B., Hunke, S., Lang, I., & Eifert, T. (2025). FAIR Digital Objects: FAIRtilizer for the Digital Harvest. *EPiC Series in Computing. 105*, pp. 284–274. EasyChair. doi:10.29007/hfzk

8 Authors' biographies

Dr. Marius Politze is head of the department "Research Process and Data Management" at the IT Center of RWTH Aachen University. Before that he held various posts at the IT Center as software developer, software architect and as a teacher for scripting and programming languages. His research focuses on Semantic Web, Linked Data, and architectures for distributed and service-oriented systems in the area of research data management. *(CRediT: Project administration, Software, Supervision, Conceptualization, Writing – original draft)*

Bernd Decker is deputy head of the Department "Process Management and Digitalization in Learning & Teaching" at the IT Center of RWTH Aachen University since 2011. From 2006 to 2009, he worked at the IT Center as a Software Developer, and since 2009 he has is leading the development group. His work focuses on IT solutions for processes in the fields of Learning Management Systems, E-Services, and Generative AI.*(CRediT: Project administration, Software, Writing – original draft)*

Uta Christoph is deputy team lead of the group "Process and Application Development for Teaching" at the IT Center of RWTH Aachen University since 2018. She has worked at RWTH Aachen University as Project Manager and Software Developer since 2014. From 2012 to 2014 she was an international consultant for airport and cargo process optimization with Inform GmbH. Her work is now focused on the IT support of the quality management and accreditation process of RWTH Aachen University. *(CRediT: Conceptualization, Investigation, Writing – review & editing)*