Data Science as a Service

Prototyping an Integrated and Consolidated IT Infrastructure Combining Enterprise Self-Service Platform and Reproducible Research

Hans Laser
Center for Information Management
Hannover Medical School
Hannover, Germany
e-mail: laser.hans@mh-hannover.de

Steve Guhr
NetApp Deutschland GmbH
Berlin, Germany
e-mail: steve.guhr@netapp.com

Jan-Hendrik Martenson
NetApp Deutschland GmbH
Hamburg, Germany
e-mail: jan-hendrik.martenson@netapp.com

Jannes Gless
Center for Information Management
Hannover Medical School
Hannover, Germany
e-mail: gless.jannes@mh-hannover.de

Branko Jandric
Center for Information Management
Hannover Medical School
Hannover, Germany
e-mail: jandric.branko@mh-hannover.de

Joshua Görner
Airbus Operations GmbH
Hamburg, Germany
e-mail: joshua.goerner@gmail.com

Detlef Amendt
Center for Information Management
Hannover Medical School
Hannover, Germany
e-mail: amendt.detlef@mh-hannover.de

Benjamin Schantze
NetApp Deutschland GmbH
Hamburg, Germany
e-mail: benjamin.schantze@netapp.com

Svetlana Gerbel
Center for Information Management
Hannover Medical School
Hannover, Germany
e-mail: gerbel.svetlana@mh-hannover.de

***Abstract*–A data scientific process (e.g., Obtain, Scrub, Explore, Model, and iNterpret (OSEMN)) usually consists of different steps and can be understood as an umbrella for the combination of different most modern techniques and tools for the extraction of information and knowledge. When developing a suitable IT infrastructure for a self-service platform in the academic environment, scientific requirements for reproducibility and comprehensibility as well as security aspects such as the availability of services and of data are to be taken into account. In this paper, we show a prototypical implementation for the efficient use of available data center resources as a self-service platform on enterprise technology to support data-driven research.**

Keywords-data science as a service; reproducible research; enterprise information technology; research data infrastructure; self-services; data science platform; cloud infrastructure.

I. INTRODUCTION

One of the most important aspects of building service portfolios in the company is to make them as simple and usable as possible for the end user [1].

Data centers at German universities are increasingly confronted with challenges in the areas of availability and operational security, data privacy and IT security, operating costs and use of cloud services, data management and access to high computing capacity, increasing standardization and consolidation of IT systems [2]. The availability of services and especially of data requires a corresponding infrastructure. Services are increasingly subjected to risk classification in order to define how the operation of university IT must be ensured. The technology used must allow to scale out (horizontally) performance of a platform, because scaling up a platform is limited to hardware resources. Today, virtualization technology is often used to compress computing power and at the same time to have the flexibility to move parts of the computing nodes and redistribute the load in case of a failure. Hardware maintenance does not necessarily result in the unavailability of a service, as it can usually be moved without interruption to another node in the data center cluster or even the cloud [3]. This infrastructure is supplemented by container technology as a further step in service encapsulation. At the same time, similar demands are made on the availability of data. Redundant Arrays of Independent (Inexpensive) Disks (RAID systems) combine performance, reliability and scalability but show weaknesses in the speed of recovery [4]. Monitoring the health of IT infrastructures with suitable tools is indispensable for professional operation [5].

The main challenge is to "transfer" (integrate) these well-known tools and solutions into a scalable and powerful platform that can be equipped with enterprise technology to ensure service level agreements (SLAs) from a centralized enterprise information technology [3].

Users of documented services are provided with more specific minimum and maximum performance measures, such as quality, punctuality or cost of a service, through the SLAs and can adapt and understand the expectations and limitations of a service accordingly [6].

The process of combining or connecting different systems and individual software applications to form a common functional system with comprehensive functionality to increase user acceptance and customer satisfaction is also known as system integration. IT service providers have an interest in the continuous improvement of product and service quality [7]. The added value for the company can be increased by improving service response times, reducing the costs for the operation of IT infrastructure, and lowering operating costs by intelligently linking the IT systems used [8].

A. *The Data Science Process*

To obtain information (e.g., based on patterns) for relevant business decisions from data of heterogeneous data sources, a classical multi-stage process for data preparation and analysis is used, the so-called data mining process [9]. Data science, on the other hand, can be understood as an umbrella for the combination of various state-of-the-art techniques for the extraction of information and knowledge (so-called insights) to develop data-based applications and, thus, automating processes. One approach to describe the individual steps for the data science process is Obtain, Scrub, Explore, Model and INterprenting (OSEMN) [10]. In the Obtain step, for example, query languages are required for databases that can be extracted in various formats. Python [11] and R [12] encapsulate the otherwise heterogeneous data query tools (e.g., Structured Query Language (SQL), eXtensible Markup Language (XML), Application Programming Languages (API), Comma Separated Value (CSV), Hybrid File System (HFS)). Classic database techniques such as Extract Transform Load (ETL) process can be used in the cleanup step (Scrub). Computer languages like Python and R or application suits like SAS Enterprise Miner [13] or OpenRefine [14] can also be used to transform data. To examine the data (Explore) languages like Python or R specialize in particular appropriate libraries (e.g., Pandas [15] or Scipy [16]). In this step, however, familiar players from the business intelligence world (e.g., Rapid Miner [17] or KNIME [18]) can also be found for data-wrangling. To build a model, there are again specialized Python libraries like “Sci-kit learn” [19] or CARET [20] for R. Other tools like KNIME or Rapid Miner find reuse in this step as well. Finally, for interpreting the model and the data as well as evaluating the generalization of the algorithm, tools for data visualization are reused (e.g., matplotlib [21], Tableau [22] or MS Power BI [23]). In summary, it means, that for the many single steps in OSEMN, many different tools can be necessary.

An example of a platform solution that maps these process steps in a so-called pipelining functionality is the Pachyderm software [24]. Pachyderm offers components that support the developer (data scientist) with regard to data provenance in development work and analyses and can thus map a logical and chronological sequence of process steps. This platform solution offers many degrees of freedom and requires the user to have a well-developed hypothesis or data processing or data management plan and is therefore suitable for the development of concrete and declarable products [25]. However, if the workflow initially requires exploration for hypothesis generation, it may be better to use tools of lower complexity.

B. *Reproducible Research*

Scientific studies, experiments and numerical calculations can only be reproduced or reconstructed if all important steps are comprehensible [26]. The importance of reproducibility of scientific work can be illustrated by the following quotation from Jon Clearbout: “An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.” [27]

A scientist should therefore always have an interest in describing the runtime environment as transparently and understandably as possible. However, complex runtime environments are difficult to penetrate due to the sometimes high technological complexity (e.g., package dependencies). The technical reproducibility enables scientific results to be reproduced at all, but this requires a very high degree of knowledge about the method and technology used. Simplifying the technological limits can increase the practical reproducibility, i.e., the actual and problem-free repeatability of the experiment [28].

In the field of computer-based data-driven science (data science), researchers today often use free and open-source tools and libraries [29]. The reproducibility and repeatability of research results and the description of the specific runtime environment in which the results were generated are not described in the respective publications or only in text form [29] [30]. An important factor in the publication of scientific work is the reproducibility of the research results [30] [31]. It is therefore necessary that deterministic environments are available to a data scientist. Determinism can be spoken of when all events, especially future events, are clearly defined by preconditions. In other words, development environments are specified accordingly and work as expected, although the methods or algorithms used cannot deliver deterministic results [32].

In data-driven research projects, the application of appropriate data management procedures helps to maintain the data integrity of digital data generated in the research process ("research data"). The preservation of data integrity in the data flow can be maintained by documenting the data origin and applied transformations during the research process. Data is considered reliable if results and errors in the data creation and data analysis process can be reproduced through traceability (good data provenance) [33].

Data provenance is encapsulated by so-called research data management, among other things. Research data management includes all measures to ensure the usability and reusability of research data before, during and after the research project [34]. Systematic representation of these points in the project life cycle is a Data Management Plan (DMP), which describes the data and how the data is processed in the project [35]. A data life cycle illustrates the steps from collection to re-use (creation, preparation, analysis, archiving, access, re-use) [36]. There are more complex models, such as the Curation Lifecycle Model (DCC), which describes various fields of activity in the preservation and maintenance of data [37]. A DMP is required by funders when submitting a proposal (see DFG Form for the continuation of a Collaborative Research Center, 1.4.3, [38]) and serves to ensure effective research data management and long-term usability of the data [34]. Various retention periods apply in order to preserve the reusability of research data. According to good scientific practice, primary data should be stored permanently in research institutions for at least ten years, together with clear and comprehensible documentation of the methods used (e.g., laboratory books) (see Recommendation 7 [39]).

In addition to the requirements arising from research data management, the data principles published in 2016, which define the basis for research data and research data infrastructures to ensure sustainability and reusability, must be taken into account [40]. It was defined by researchers, financiers, publishers and university representatives to increase the reusability of research data (FORCE 11 group). Scientific data should therefore be searchable, accessible, interoperable and reusable. The FAIR data principles can be applied to the entire data life cycle and, as an extension to research data management, provide a collection of best practices for sharing data under ethical and contractual conditions (copyrights, intellectual property rights, etc.).

Highly simplified, data and services should be stored in central data repositories using appropriate metadata (F), taking into account aspects of long-term archiving (A), and should be able to be exchanged and interpreted (semi-)automatically (I) and thus be comparable and reusable (R). If research data cannot be published due to legal requirements, the FAIR principles provide procedures for publishing a description to make the underlying data more understandable. Metadata on machine or human-readable interpretability facilitate comprehensibility and data processing (see Dublin Core Metadata Initiative, Data Documentation Initiative, etc.), open data formats facilitate interchangeability and reusability, metadata on privacy and copyright regulations facilitate accessibility, persistent identifiers assist in finding and easily accessing the information, the indication of licenses (e.g., Creative Commons, etc.) specify the type of usability of the data [41]. Numerous tools support the work with FAIR-Data [42].

In the context of data-driven science, several concepts that place demands on an ecosystem of IT systems must therefore be considered. Following the OSEMN process, the data life cycle of research data management and the FAIR data principles, a schematic summary is shown in Figure 1.

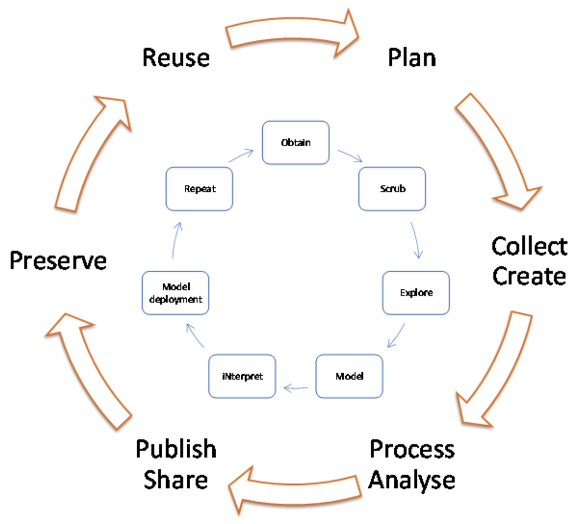

Figure 1. A schematic summary of OSEMN process, the data life cycle of research data management and the FAIR data principles (Source: own illustration).

In addition, depending on the subject area, further guidelines are to be considered. For the secondary data analysis of health data, these are, among others, the "Good Practice in Secondary Data Analysis (GPS)" [43] and the "Good Clinical Practice of the International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH E6 GCP) [44][45].

IT operators at research institutions should therefore work together with researchers and libraries on IT infrastructures to support the above points. Results of systems research show that open source tools in particular are suitable for the requirements of reproducibility. Although Docker was introduced primarily for business needs and the isolation and deployment of web applications in so-called containers, it provides solutions for virtualization, platform portability, reuse, sharing, archiving and versioning [17] for the scientific community.

The use of tools such as JupyterNotebooks (jupyter.org) enables semantically interoperable publishing of program code, including through the use of the IPYNB format [29]. JupyterNotebooks supports workflows in scientific computing, astrophysics, geology, genetics and computer science [29]. Various applications and programming languages (e.g., Python, R) provide interfaces to JupyterNotebooks [30][46]. Jupyter collects many valuable tools that are needed in the steps of the OSEMN process model.

C. *The aims of the project*

The aim of this project is to create an easy-to-maintain and cost-effective consolidated IT infrastructure to support data-driven research and implementation in the existing data center infrastructure at the Center for Information Management (ZIMt) at Hannover Medical School (MHH). The requirements for IT systems of the known process models, research data management, FAIR and the operation of applications on enterprise level (such as data security and system recoverability) and numerous other standards, guidelines, directives and recommendations (such as Good Research Practice, Good Practice for Secondary Use of Data, etc.) have to be met.

The ZIMt centralizes operative systems and is a service provider especially for the areas of research and teaching, clinic and administration. IT services (applications) are used in clinical operations to optimize clinical processes and legal documentation and place high demands on system availability and fail-safe IT services. This has an impact on the processes and IT systems of the computer center. In addition, simple interfaces are provided for end users (nursing, doctors, administration) for problem presentation and reporting, which enable centrally controlled fault clearance via an IT service desk. In order to guarantee interference suppression, high demands are placed on the standardization of IT processes and system documentation. The ZIMt operates a class 3 [47] data center at the MHH to guarantee these requirements and is certified according to ISO 9001:2015. In the area of research, however, the IT process landscape is sometimes disruptive, as rapidly changing requirements and moving targets are sometimes necessary to achieve the research goal. The centralization of applications to support the scientific sector is a strategic goal of the MHH.

We have defined three main areas of focus which are to be given special consideration in this proof of concept draft: (1) usability to evaluate the integratability of the IT solution into the system landscape established at the MHH and the working environment familiar to the end user; (2) disaster recovery to evaluate the recoverability of the proposed solution. By integrating a system into the MHH environment, the system has to meet special requirements (front-end branding to maintain corporate identity and security aspects). The user administration and access authorization (3) should be used centrally via an existing directory service ("Active Directory") and security groups defined therein, and must therefore be evaluated. Requirements on the usability of the service can in turn have an influence on the security factors.

These dependencies should therefore be emphasized in this concept. For easy control of used storage resources, the available storage system of the data management provider NetApp [48] was applied. Available interfaces should be used and thus not require any additional effort in the management and monitoring of the system for the IT operator.

This paper is an extended version of our previous work [1] and the further course of this paper is structured as follows. In Section II we describe the methods used to meet the above challenges. In Section III we describe the results achieved in relation to the main topics. Section IV concludes this paper, addressing open questions and the next steps.

II. METHODS

As an interactive shell for various programming languages, JupyterNotebooks is provided as development environments at the MHH. The solution is browser-based and the end user does not need to install any additional software on his device [49]. The open source environment JupyterHub Notebook Server is used to operate JupyterNotebooks in the data center. It enables users to access computer environments and resources without bothering them with installation and maintenance tasks.

Standardized environments (the Docker software/binaries that run on many different operating systems today) built for containers are suitable for using individual application manifests in different locations (e.g., locally as well as in a public cloud infrastructure provided by Google, Amazon Web Services (AWS) or Microsoft Azure) without having to change the code. JupyterHub uses Docker as the basis for the deployment of JupyterNotebooks. The Jupyter Docker Stacks project [50] provides standardized JupyterNotebook environments for various applications using Docker images, including preconfigured environments for use in data science. For special requirements of the development environment, additional images can be offered that can be individually adapted to the user's needs.

In Kubernetes, several containers (e.g., Docker Containers) with common memory and network (common context) can be defined and delivered in a so-called capsule ("Pod") as a coherent structure [51]. For more design flexibility, a hypervisor (VMware) was used to provide the hosts for container orchestration based on Dockers and Kubernetes. Snapshots on VMware enable point-in-time copies of virtual disks can be used to switch to a virtual machine state at an earlier time.

Terraform and Ansible are fully automated. Terraform is an open-source tool that offers the possibility to describe infrastructure configurations programmatically as code (CaaS [52]) (Hashicorp Configuration Language) [53]. As an open-source tool, Ansible provides automation tools for orchestration, general configuration (e.g., software distribution) and administration of IT infrastructures [54]. Terraform creates the virtual machines within the hypervisor, Ansible takes care of the installation of the packages and the configuration of the hosts (configuration management). To avoid configuration inconsistencies, the DevOps (Software Development and Information Technology Operations) paradigm is considered an "immutable" infrastructure where every change in the ecosystem leads to a completely new deployment of the entire stack [55].

A JupyterHub is a multi-user server for JupyterNotebooks, consisting of several applications that provide different services (hub, notebooks, proxy, authenticator, spawner), which allow secure access to central computing environments and resources.

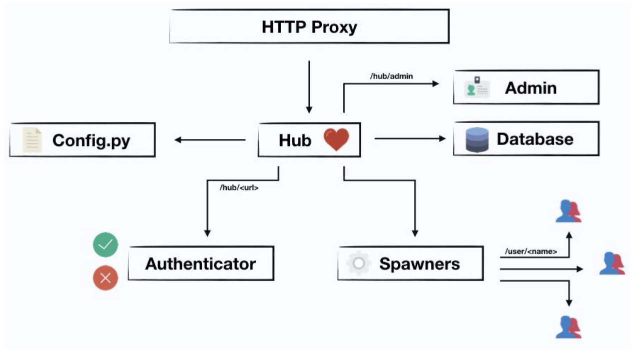

Figure 2. Overview of JupyterHub components [56]

The most important components and their interaction can be briefly explained using the JupytherHub architecture (see Figure 2). The JupyterHub proxy forwards the user to the JupyterHub or to the user's existing JupyterNotebook, depending on the user's sign-on status (SIGNED IN/OUT USER). Existing JupyterNotebooks are managed in the Hub and after successful authentication new JupyterNotebooks as well as a defined user storage (Pods + Volumes) are provided and registered in the proxy. The Hub Container of the JupytherHub as a central component contains three services (Authenticator, User Database, Spawner), which on the one hand can be adapted to the personal requirements and on the other hand are particularly worth protecting [56]. The so-called "Authenticator" authenticates the users against a directory service, in our case a Microsoft Active Directory using the Lightweight Directory Access Protocol (LDAP), and provides the required input mask as a web page. The "User Database" stores login and JupyterNotebook information of the respective users. This data is required for operation and recovery [57]. It is recommended to replace the standard database (SQLite) in a productive environment with a classic relational database management system (RDBMS) such as PostgreSQL or MySQL. The spawner provides the notebooks, is able to communicate with the Kubernetes API and creates ("spawned") Docker Containers. The spawner can be parameterized, so that resources can be limited or a special image can be passed to create the containers via Kubernetes [58][59].

For the proper implementation and configuration of all components, in addition to a manual deployment, helm as used, a package manager for easy installation, publishing, administration, updating and scaling of preconfigured Kubernetes applications. All configurations and communications between the containers can be predefined, as well as the accessibility of the end users from outside. In this project we used a predefined Helm-Chart [60] for the deployment of JupyterHub on Kubernetes. Any changes to the predefined default values can be overwritten by a configuration file during deployment and thus be individually adapted to the requirements in the respective system environment.

Since Docker Containers do not persist data after their life cycle has ended, the storage of configuration and user data must be guaranteed. You must therefore mount persistent volumes to retain the data beyond the life cycle of the container. For this purpose, Kubernetes offers Persistent Volumes, which are usually stored on the nodes of a Kubernetes cluster [61]. In the MHH data center, a high-performance and highly available NetApp storage system is used as the central storage system for consolidating and storing data. To prevent data from being stored on individual distributed devices, the data from the Kubernetes containers should be persisted on this central storage. For this purpose, persistent volumes were provided automatically and centrally via an open source for Kubernetes from NetApp, Trident [62]. Trident offers an ideal interface between persistent volumes and the containers or Pods for this purpose. For each Pod, a separate volume was created on the storage cluster, which by default is only provided for one Pod (e.g., JupyterNotebook). Using so-called storage classes, NetApp storage can map various guidelines for quality of service level or back-up guidelines, among other things, in order to differentiate resource allocation between storage services for students (short availability, balanced performance, daily back-ups) and researchers (long availability, high performance, high back-up frequency).

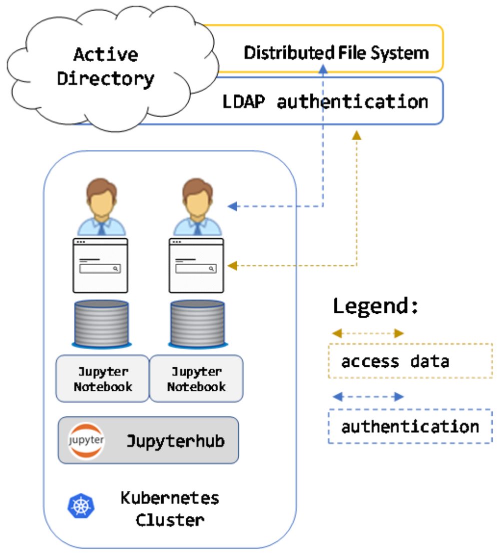

Figure 3. Sketch access to network drive on DFS from a JupyterNotebook context (Source: own illustration).

The workstations at the MHH are equipped with Microsoft Windows by default. The operating mode is strongly influenced by the look and feel of the graphical user interfaces, and project documents are stored on network drives provided by the data center. Microsoft enables directories distributed on different data storage devices in the network to be combined into directory structures via the Distributed File System (DFS). The Server Message Block (SMB) protocol is used to map the file system authorizations with authorization objects of the Active Directory in the network (via DFS) (see Figure 3) [64]. In JupyterHub the user's network drives are not available in raw state, but he must be able to store raw data, the generated program codes and result files from and to the network drive where the central access (also of the team members) takes place.

The used JupyterHub container is based on a Linux operating system (Ubuntu). A user can mount SMB shares in the available directory structure under Linux using the LinuxCIFS utils package [64]. For the use of LinuxCIFS utils increased system rights are required. This requires a corresponding implementation within the container, but Docker also implements standard security rules and thus, for example, prohibits the execution of the command for mounting network drives in the default settings, which in Kubernetes is done via the so-called "privileged mode" [65] [66]. For security reasons, applications should only be given the most necessary privileges (see chapter 6 [67]). To enable the privileged mode, the corresponding configuration parameters were transferred via the Helm-values-file (config.yaml). The LinuxCIFS utils are not included in the used Dockerfile [68] of JupyterNotebook of this JupyterHub Helm chart. We have manipulated the Dockerfile accordingly, and integrated the LinuxCifs utils. In addition, the user under which the JupyterNotebook is executed ("jovyan") needs increased rights of the superuser (so-called "root") to execute the command. The assignment of the execution rights is done via the "sudoers" file. The user jovyan has only been granted rights to execute the following modules (mount/umount). Then the Docker image was built with this customized Dockerfile and deployed on each Kubernetes worker node. The JupyterHub Helm chart with a customized helm-values-file (config.yaml) was installed to use the new Docker image [69]. To give the end user the familiar look and feel of the corporate environment, it is necessary to customize the application according to the corporate design of MHH. On the log-in page, the user receives the company logo so that there is direct recognition value to an in-house application. For this purpose, an adapted version of the HTML files for the log-in page of JupyterHub was provided on the Kubernetes workstations. The files are located in the container of JupyterHub in the Unix path “/usr/local/share/jupyterhub/templates” and were adapted to a path on the Kubernetes-Worker using helm-values. Kubernetes controls this process automatically. The log-in pages can thus be adjusted at runtime.

Even though we are in a "proof of concept" phase of the project, we wanted to integrate authentication methods from the beginning to control access to the platform while achieving security compliance. At the MHH, a local security area for managing objects (e.g., user names, computers, printers, etc.) is implemented as a domain via Microsoft Windows Active Directory. To centralize the administration of user IDs using Role-Based Access Control (RBAC), authentication to the Active Directory was accessed using JupyterHub's Lightweight Directory Access Protocol (LDAP) implementation. A further step towards simplifying the login to a service, centralizing authentication and ensuring the company's password policies was the integration of user authentication via LDAP. For this purpose, the section for the Authenticator in the Helm-values was adapted. A special security group was defined in the Active Directory to restrict the user circle and to allow a control of user releases for this service.

The user can decide via different spawners which Docker image and thus which standard runtime environment and amount of resources should be provided. This was also realized via the Helm-values. With preconfigured spawners, the user does not have to configure the runtime environment every time he/she starts the environment, e.g., to get an integration of special packages. A selection of preconfigured Dockerfiles is available on the Docker-Hub [70].

Due to the central storage of the runtime environment and the strict alignment of the project to CaaS, a quick recovery of the service is easily possible. The following measures were taken to achieve this:

1. creating the Kubernet cluster and separate needed virtual machines via Vagrantfiles
2. configuration of machines using Ansible Playbooks
3. restore the Kubernetes database including Trident configuration [19]
4. restore JupyterHub using Helm-chart and persistent volumes

JupyterHub is a central element in the OSEMN process model. It can be used for data preparation, feature extraction and model programming in the steps Scrub, Explore and Model. We recommend the use of the openData Platform CKAN [71] for the acquisition and administration of data sets. This modern UI enables easy navigation and search for available data sets from other or own projects using suitable metadata in accordance with the FAIR principles (see, e.g., [72]). CKAN is thus a possible technological component in the Obtain process step. CKAN also offers the possibility to store the files in an object storage via an Amazon S3 interface [73]. During the entire data science process, it may be necessary to load, process or store data or artifacts from different sources. Especially for different "unstructured" data, the use of object storage as data storage is recommended, since the administration of the objects by means of metadata provides a higher flexibility and context description of the data sets than when storing them on a common file system. To manage the contents in object storage, we recommend the use of Minio. Minio implements interfaces to Java, Go, Node.js, Python and .NET [74] and is therefore well suited for the most common programming languages in the Data Science environment [75]. For structured data and using the Python libraries pandas [76] and SQLAlchemy [77] a database management system

(DBMS) is recommended. An OpenSource solution of a DBMS is Postgres [78]. In the steps Explore and Interpret it is helpful to create visualizations of the data. A suitable open source tool for this can be Superset [79]. Superset is a dashboarding tool that is easy to use with little technical know-how and offers a variety of ready-made and interactive visualizations. As interfaces with databases Superset implements the well-known JDBC or ODBC drivers [80].

The technological components used enable the monitoring and evaluation of the health status of services using various metrics. The number of running Pod's under Kubernetes gives information about the indirect number of users, the users logged on the system. The utilization of individual Pods is possible via the Kubernetes service "heapster" [81]. The latency is a measure to evaluate the reaction time between application and client (end user). A monitoring of the response time can be realized by different methods. In our case, the central IT monitoring software "Checkmk" [82] is used, which allows to monitor different metrics of a device. This way, besides latency, other essential metrics such as memory usage and resource utilization (CPU/RAM) can be monitored. With the help of this monitoring it is possible to react proactively to upcoming problems. In addition, by storing the monitoring data, long-term analyses and trends can be identified, which can be used to plan the expansion of the environment.

Kubernetes exposes interfaces to Kubernetes management and cluster control in its own network segment. To prevent users from the corporate network from accessing these management interfaces, the Kubernetes Cluster was configured with a separate network having its own IP range. Via a so-called reverse proxy [83] we enable services from the network segment of the Kubernetes Cluster to be exposed in the corporate network (separate segment). The reverse proxy accepts requests from the company network and forwards them (depending on the given address), e.g., into the network of the Kubernetes Cluster. In this way this proxy provides, e.g., for JupyterHub centrally the URL to access the platform and extends the environment by an encrypted data transfer between end user and the JupytherHub Proxy, using the Transport Layer Security (TLS) [84]. When providing Pods on Kubernetes, services are bound to the IP address of the possible Kubernetes node via ports. By default, these IP addresses and ports are assigned dynamically at the time of provision [85]. We have assigned each service via a specific port according to the LoadBalancer principle [86][87]. This way the reverse proxy can reach the service at any time in the Kubernetes Cluster. Kubernetes takes care of the failover thanks to the integrated High Availability functions using LoadBalancer.

III. RESULTS

Based on the methods described in II, an ecosystem consisting of the proposed software components for mapping services was implemented. Using JupyterHub as an example, the levels at which such a service must be integrated into an enterprise in order to meet the requirements placed on an IT service provider were illustrated.

The operator of the infrastructure (ZIMt) achieves a reduction of workload through the chosen reference architecture (see Figure 4) by automating the provision of resources for the users (researchers) and minimizing the effort to provision resources for research purposes.

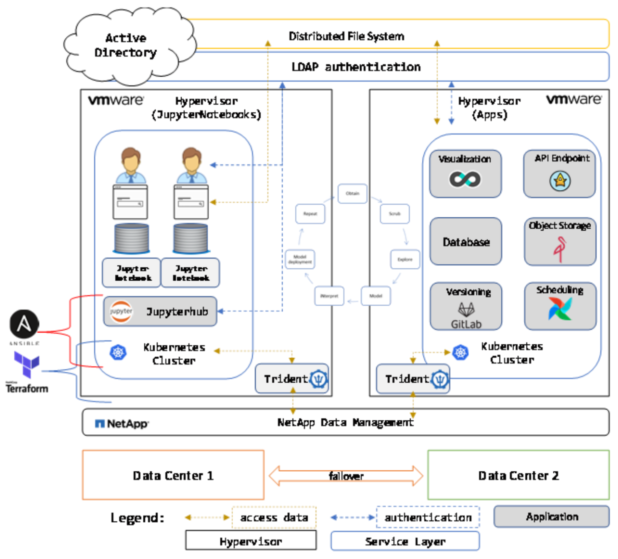

Figure 4. Prototypical architecture for deploying JupyterNotebooks on enterprise technology. To the centrally indicated OSEMN process, the Kubernetes node for the JupyterHub infrastructure is shown on the left. On the right, another Kubernetes node is shown with additional exemplary tools to support the OSEMN process. The components are in failover to a second data center (see bottom) (Source: Own illustration).

The prototypically implemented infrastructure enables the end user (students, scientists) to easily use JupyterNotebooks. With the Docker-based approach, the description of the runtime environment required for the research approach can be fixed using the Docker-specific tagging facility and stored in a manifest for publication in a comprehensible and interoperable manner [30].

JupyterHub allows users to interact with a computing environment via a web page. Since most devices have access to a web browser, JupyterHub makes it easy to deploy and standardize the computing environment to a group of people (for example, a class of students or an analysis team). Additional tools (CKAN, Postgres, Minio, Superset) are also accessible via a web interface and provide additional programming interfaces that can be addressed in JupyterNotebook. Furthermore, it could be shown that Docker images of the JupyterNotebooks, especially adapted to the specific requirements of the company, could be created and made available for selection via the JupyterHub Spawner (Basic and Data Science). Additional libraries or tools, which must not be included in the standardized environment, can be flexibly installed in a separate area within the current runtime environment. JupyterNotebooks thus offer the possibility to use several tools without changing the environment. Requirements for different tools, such as those needed in processes like OSEMN, can be mapped with the software products mentioned above (see Table I).

TABLE I. MAPPING OSEMN PROCESS PHASES AND PROPOSED TECHNOLOGY

OSEMN Phase	Domain	Proposed Technology
Obtain	Data Search	CKAN
	Data I/O	Postgres
		Minio
Scrub	Data Transformation	Jupyter
	Data I/O	Postgres
		Minio
Explore	Pattern Finding	Superset
	Feature Extraction	Jupyter
	Data I/O	Postgres
		Minio
Model	Modeling	Jupyter
	Data I/O	Postgres
		Minio
iNterpret	Review	Superset

If the researcher chooses a working environment based on JupyterNotebooks, necessary work steps and results can be stored together with the notebook [29][46]. The basic requirements for the implementation of requirements from research data management and the FAIR principles can be fulfilled. A notebook acts as a laboratory book and describes the steps of data processing. The GitLab also provides a history function, so that the researcher can ensure data provenance in the research project. The isolation of a researcher's specific work area can therefore be achieved by using container technology. Kubernetes offers the possibility to consolidate central computing resources in a data center and to use them efficiently due to the integrated load distribution and error bypass.

It could be shown that already available IT infrastructure could be sensibly integrated (including storage systems, hypervisor infrastructure, management, monitoring). The connection of the components to the company's own IT monitoring system consolidates the various metrics in one place and facilitates the management of the prototypically implemented infrastructure.

Since JupyterHub is provided via Dockers, branding requirements to maintain corporate identity can be easily met. It was shown that the login page of JupyterHub could be adapted to the corporate design of the company and, among other things, be branded with the company logo. By using the existing Active Directory, user access can be managed centrally. Authentication via LDAP simplifies the login to the system, since no separate access data has to be maintained and incorrect logins can be registered and used to block the user account in case of attempted misuse.

Each time a user logs on to the start page, the system checks whether the user already owns a JupyterNotebook created in the past. In this case he can be redirected to his previously created environment. Otherwise, a new notebook (in the form of a new container) is created and new storage space is provided (because this user did not exist before). By providing the required storage space at runtime, resources can be provided centrally and efficiently. The persistence of research data outside the Docker Container runtime on the existing enterprise storage system could be solved efficiently by using Trident (see Figure 4). This means that the processing steps required during the process (including temporary ones) can be performed on a high-performance storage system that can guarantee the persistence of research data in any case. Nevertheless, the user is able to make the final script files and data of the project available to other members of the department via the mountable department drive. The department drive (DFS) (see Figure 4) can thus be reused in its original function and offers researchers without technical affinity the possibility to access the project data via their usual working methods.

IV. DISCUSSION

In this paper we show a prototypical implementation for the efficient use of available data center resources as a self-service platform on enterprise technology to support data-driven research.

Although the OSEMN process is a suitable, easy-to-understand reference, there are some extensions that are proposed below. A major drawback of the OSEMN process is that it is understood as a linear, aperiodic process. Compared to other established process models for data analysis/data science, such as Knowledge Discovery in Databases (KDD) [88] or Cross Industry Standard Procedure for Data Mining (CRISP-DM) [89], the knowledge gained is not played back and (at least formally) no iterations take place. However, this re-iteration is a decisive step, since many projects are more successful due to their exploratory character [90], if they are carried out in short iterations. In the course of these cycles many of the tasks are repeated, such as data cleansing or training of models. In order to use the available resources as effectively as possible, it is recommended to aim for the highest possible degree of automation (extension by the Repeat component). Possible tools for this would be Apache Airflow [91] or the Python library Kedro [92]. Another crucial step, which is included in the CRISP-DM model in contrast to OSEMN, is the deployment of the developed (ready-to-use) model (see "Phase 6 - Deployment" in CRISP-DM). This allows the trained models to be decoupled from the underlying technology (e.g., Python, R, Julia) and made available to a wider audience via standardized web interfaces (REST via HTTP). Examples are Python libraries like Flask [93] or HUG [94]. When providing models, the aforementioned automation aspect has quality assurance features. The "decay" of a model can be detected and corrected proactively by regular and automated testing of the endpoints (see Model Decay & Concept Drift [95]). We therefore propose to extend the OSEMN process model after the iNterpret step by the steps Model Serve/Deploy ("Mo") and Repeat ("Re") transversal to OSMEN (see Table II).

Kubernetes was used as an open source solution to orchestrate, automate and fulfil these high availability requirements for the container-based infrastructure [96]-[98]. It is a widely used and proven technology for providing services like Jupyter. Since Kubernetes is designed to host a huge number of containerized applications with minimal overhead, it is perfectly suited for many JupyterNotebooks and other potential applications within the science ecosystem [99][100].

TABLE II. MAPPING OSEMN PROCESS PHASES AND PROPOSED TECHNOLOGY

MoRE OSEMN Phase	Domain	Component	Proposed Technology
Model Serve & Deploy	Deployment	API End Point	API Star
	Tracking	Usage Monitoring	CheckMK
		Model Monitoring	Superset
Repeat	Automation	Scheduling Engine	Airflow

For the prototypical implementation of this infrastructure a Kubernetes master with two Kubernetes nodes ("Worker") was used. For productive operation, at least two Kubernetes Masters should be used in order to meet the requirements for fail-safe operation. In the event of a disaster recovery scenario and the loss of the entire Kubernetes cluster, the storage volumes provided via Trident must be reconnected manually. An automatism for recovery procedures would still have to be created. In an emergency, the administrator can migrate the contents of the corresponding volume using an NFS interface.

Docker Containers are more convenient to implement, easier to manage, minimize the overhead of resource usage, and are therefore more efficient than traditional virtual infrastructures [99]. The provision of environments (e.g., by containers) in the academic sector can be very large, thus increasing the burden on the operators and maintainers of the environment. Automating the deployment and orchestration of the environment is strongly recommended. Running applications in containers does not automatically solve the challenge of protecting these applications from outages (such as hardware failures or resource bottlenecks on the one server we are working on). Even if containers are encapsulated on an operating system, there may be problems with the underlying host on which the container is running - therefore an additional software layer is required to take care of resource planning and availability of any services.

The use of the predefined configurations of the JupyterNotebooks is initially limited by the Docker Images. If the service is used for a longer period of time, it will become apparent whether the provision of additional images makes sense.

The authorization of the user in the JupyterNotebook Docker Container to mount SMB network drives by sharing via the sudoers file and the activation of the increased privileges in Kubernetes inevitably leads to serious security vulnerability. The user would be able to provide his own file system and replace the sudoers file with a specially manipulated file. The consequence would be system administrator privileges. The solution used must therefore be adapted taking security aspects into account. For example, Kubernetes could be enabled via the vSphere API [https://github.com/kubernetes-sigs/cluster-api-provider-vsphere] to provide each JupyterNotebook-Pod in an isolated virtual machine, which would result in a stricter isolation of the Docker Containers from each other.

In order to make it as easy as possible for end users to transfer the artefacts created in the JupyterNotebook to the project repositories on the network drives, the end user should be able to integrate the network drive into his JupyterNotebook environment. We expect a higher user acceptance despite the use of new technology / application.

A Pod (container), in which a JupyterNotebook is running, expires after a certain time without activity (max idle time) [101], so that the resources can be released again and used for other users / Pods (downscaling). If a Pod expires, changes made at runtime are also discarded, since they are not part of the Docker image that is called by JupyterHub every time a Pod is initialized. The user must therefore remount his network drives after such a reset.

Existing and established authentication methods such as OAuth [102] or OpenID [103] offer additional flexibility, as users outside of the Active Directory could be integrated. The necessary components were not available during the project. However, these security concepts will be considered in later phases of the project after the validation of the first thesis (if this software stack is suitable for the use case at all).

Despite monitoring and a high degree of automation of the individual system components, errors can occur during operation. These can be of different nature, but can be roughly divided into logical and physical errors. Logical errors in data lead to inconsistencies and physical (hardware) errors are associated with data loss. As a countermeasure, the system components are protected by creating backups. For this purpose, system copies - so-called "snapshots" - are created at regular intervals by the hypervisor or storage technology used, which can be accessed as required. If the JupyterHub internal database is lost, the connection to the Pod and thus to the individual runtime environment is lost and must be restored.

The presented proof of concept could demonstrate the feasibility of IT operations by combining common data science tools with the enterprise architecture. In the next development stages, tools such as Airflow [91] or Pachyderm [24] (as a platform solution) for pipelining and automation can be used in addition to JupyterNotebooks in connection with machine learning. These tools could support process models like OSEMN as well as aspects of reproducibility and reusability. Integration of tools specifically for big data use cases is not recommended, as these may require special ecosystems (see [104]). More often than not, the end user can already interact with available big data platforms at any time using the programming languages available in JupyterNotebooks.

REFERENCES

[1] S. Guhr et al., “Data Science as a Service - Prototyping for an Enterprise Self-Service Platform for Reproducible Research“, IARIA - The Fifth International Conference on Fundamentals and Advances in Software Systems Integration (FASSI 2019), 2020

[2] Landesarbeitskreis Niedersachsen für Informationstechnik/Hochschulrechenzentren, “Landes-IT-Konzept Hochschulen Niedersachsen”, 2018, https://www.lanit-hrz.de/fileadmin/user_upload/Landes-IT-Konzept_Hochschulen_Niedersachsen_2019-2024.pdf, last accessed 2020/02/27

[3] V. Kale, “Big Data Computing: A Guide for Business and Technology Managers”, Chapman and Hall/CRC, 2016, ISBN: 978-1498715331

[4] W. C. Preston, “Does Object Storage kill RAID?”, Storage Switzerland, 2016, https://storageswiss.com/2016/02/09/does-object-storage-kill-raid/, last accessed 2020/02/27

[5] J. Hernantes, G. Gallardo, and N. Serrano, "IT Infrastructure-Monitoring Tools," IEEE Software, vol. 32, no. 4, pp. 88-93, 2015. DOI: 10.1109/MS.2015.96

[6] The International Foundation for Information Technology, 2009, https://www.if4it.com/SYNTHESIZED/GLOSSARY/S/System_Integration_Management_Service_Level_Agreement_SLA.html/, last accessed 2020/02/27

[7] C.-P. Praeg and D. Spath, “Quality Management for IT Services: Perspectives on Business and Process Performance (Advances in Logistics, Operations, and Management Science)”, Business Science Reference, 2011, ISBN: 978-1616928896)

[8] M. A. Vonderembse, T. S. Raghunathan, and S. Subba Rao, “A post-industrial paradigm: To integrate and automate manufacturing”, International Journal of Production Research, 35:9, 2579-2600, 1997, DOI: 10.1080/002075497194679

[9] C. Li, “Preprocessing Methods and Pipelines of Data Mining: An Overview”, CoRR, 2019, arXiv:1906.08510

[10] H. Mason and C. Wiggins, “A Taxonomy of Data Science”, dataists.com, 2010, http://www.dataists.com/2010/09/a-taxonomy-of-data-science/, last accessed 2019/07/22

[11] Python Programming Language, 2019, https://www.python.org, last accessed 2020/02/23

[12] The R Project for Statistical Computing, https://www.r-project.org, last accessed 2020/02/23

[13] SAS® Enterprise Miner™, 2019, https://www.sas.com/en_us/software/enterprise-miner.html, last accessed 2020/02/23

[14] Open Refine, http://openrefine.org/, last accessed 2020/02/23

[15] Pandas Python Data Analysis Library, https://pandas.pydata.org/, last accessed 2020/02/23

[16] Scipy, 2019, https://www.scipy.org/, last accessed 2020/02/23

[17] Rapid Miner, 2019, https://rapidminer.com/, last accessed 2020/02/23

[18] KNIME End to End Data Science, 2019, https://www.knime.com/, last accessed 2020/02/23

[19] scikit-learn: machine learning in Python, https://scikit-learn.org/stable/, last accessed 2020/02/23

[20] A Short Introduction to the caret Package, https://cran.r-project.org/web/packages/caret/vignettes/caret.html, last accessed 2020/02/23

[21] matplotlib, 2019, https://matplotlib.org/, last accessed 2020/02/23

[22] Tableau, 2019, https://www.tableau.com, last accessed 2020/02/23

[23] Microsoft Power BI, 2019, https://powerbi.microsoft.com, last accessed 2020/02/23

[24] Pachyderm - Reproducible Data Science that Scales!, 2019, https://www.pachyderm.io/, last accessed 2020/02/23

[25] A. Woodie, “Inside Pachyderm, a Containerized Alternative to Hadoop”, datamai, 2018, https://www.datanami.com/2018/11/20/inside-pachyderm-a-containerized-alternative-to-hadoop/, last accessed 2020/02/23

[26] Max Planck Society, “Rules of Good Scientific Practice”, 2000, https://www.evolbio.mpg.de/3306231/rules-of-good-scientific-practice.pdf, last accessed 2020/02/28

[27] J. B. Buckheit and D. L. Donoho, “WaveLab and Reproducible Research”, Wavelets and Statistics, pp. 55-81, 1995, DOI: 10.1007/978-1-4612-2544-7_5

[28] M. Bussonnierk et al., “Binder 2.0 - Reproducible, interactive, sharable environments for science at scale”, Conference: Python in Science Conference, 2018, DOI: 10.25080/Majora-4af1f417-011

[29] T. Kluyver et al., Jupyter Development Team, “JupyterNotebooks – a publishing format for reproducible computational workflows”, IOS Press. pp. 87-90, 2016, DOI: 10.3233/978-1-61499-649-1-87

[30] C. Boettiger, “An introduction to Docker for reproducible research, with examples from the R environment”, ACM SIGOPS Oper. Syst. Rev.. 49. 10.1145/2723872.2723882. https://arxiv.org/pdf/1410.0846.pdf

[31] Nature Editors 2012. “Must try harder”. Nature. 483,7391 Mar. 2012, 509–509.

[32] D. L. Donoho, “An invitation to reproducible computational research”, Biostatistics, Volume 11, Issue 3, July 2010, pp. 385–388, DOI: 10.1093/biostatistics/kxq028

[33] E. Boose et al., “Ensuring reliable datasets for environmental models and forecasts”, Ecological Informatics 2(3):237-247, 2007, DOI: 10.1016/j.ecoinf.2007.07.006

[34] K.F. Holmstrand, S.P.A. den Boer, E. Vlachos, P.M. Martínez-Lavanchy, and K.K. Hansen, “Research Data Management (eLearning course)”, Eds., 2019. doi: 10.11581/dtu:00000047

[35] Wikipedia contributors, “Data management plan”, Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Data_management_plan&oldid=918458183, last accessed 2020/02/29

[36] UK Data Service, “Research data lifecycle”, https://www.ukdataservice.ac.uk/manage-data/lifecycle.aspx, last accessed 2020/02/28

[37] Digital Curation Centre University of Edinburgh, “DCC Curation Lifecycle Model”, http://www.dcc.ac.uk/resources/curation-lifecycle-model, last accessed 2020/02/28

[38] Deutsche Forschungsgemeinschaft (DFG), “Antragsmuster für die Fortsetzung eines Sonderforschungsbereichs”, 2019, https://www.dfg.de/formulare/60_200/60_200_de.pdf, last accessed 2020/02/28

[39] German Research Foundation (DFG), “Safeguarding Good Scientific Practice”, 2013, DOI: 10.1002/9783527679188.oth1

[40] M. D. Wilkinson et al., “The FAIR Guiding Principles for scientific data management and stewardship”, Scientific Data 3, doi : 10.1038/sdata.2016.18, 2016.

[41] P. M. Martínez-Lavanchy, F. J. Hüser, M. C. H. Buss, J. J. Andersen, and J. W. Begtrup, “FAIR Principles, DOI: 10.11581/dtu:00000049

[42] Danish e-infrastructure Cooperation (DeiC), “FAIR for Beginners”, https://vidensportal.deic.dk/en/FAIR, last accessed 2020/02/28

[43] E. Swart et al., “Good Practice of Secondary Data Analysis (GPS), guidelines and recommendations), Gesundheitswesen

2015; 77(02): 120-126, Third Revision 2012/2014, DOI: 10.1055/s-0034-1396815

[44] International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH), “Guideline for good clinical practice”, 2016, https://database.ich.org/sites/default/files/E6_R2_Addendum.pdf, last accessed 2020/02/28

[45] G. Tancev, “An Introduction to Clinical Data Science”, 2019, https://towardsdatascience.com/clinical-data-science-an-introduction-9c778bd83ea2, last accessed 2020/02/28

[46] E. Cirillo, “TranSMART data exploration and analysis using Python Client and JupyterNotebook”, 2018, http://blog.thehyve.nl/blog/transmart-data-exploration-and-analysis-using-python-client-and-jupyter-notebook,last accessed 2019/07/22

[47] OVH SAS. 2018 “Understanding Tier 3 and Tier 4”. https://www.ovh.com/world/dedicated-servers/understanding-t3-t4.xml, last accessed 2018/09/15.

[48] K. Kerr, “Gartner Named NetApp a Leader in Magic Quadrant for 2019 Primary Storage”, 2019, https://blog.netapp.com/netapp-gartner-magic-quadrant-2019-primary-storage/, last accessed 2020/02/23

[49] J. VanderPlas, “Python Data Science Handbook: Essential Tools for working with Data”, O'Reilly UK Ltd., 2016, https://jakevdp.github.io/PythonDataScienceHandbook/, last accessed 2020/02/28

[50] Jupyter Docker Stacks, 2018, https://jupyter-Docker-stacks.readthedocs.io/en/latest/, last accessed 2020/02/23

[51] Kubernetes Authors, “Pods”, 2020, https://kubernetes.io/docs/concepts/workloads/Pods/Pod/, last accessed 2020/02/28

[52] C. de Botton, “Writing Your Code as a Service, Part I”, 2015, https://medium.com/@brooklynfoundry/writing-your-code-as-a-service-part-i-2b960c19ca6e, last accessed 2020/02/28

[53] HashiCorp, “How Terraform Works”, https://www.terraform.io/docs/extend/how-terraform-works.html, last accessed 2020/02/28

[54] Red Hat, “OVERVIEW How Ansible Works”, https://www.ansible.com/overview/how-ansible-works, last accessed 2020/02/28

[55] P. Debois and J. Humble, “The DevOps Handbook: how to create World-Class agility, Reliability, and Security in Technology Organizations”, IT Revolution Press, 2016, ISBN: 978-1942788003

[56] Project Jupyter team, “JupyterHub”, 2016, https://jupyterhub.readthedocs.io/en/stable/, last accessed 2020/02/28

[57] M. van Niekerk, “Recovering from a Jupyter Disaster”, 2019, https://medium.com/flatiron-engineering/recovering-from-a-jupyter-disaster-27401677aeeb, last accessed 2020/02/28

[58] jupyterhub_config.py, https://github.com/jupyterhub/jupyterhub/blob/master/examples/spawn-form/jupyterhub_config.py, last accessed 2020/02/28

[59] Project Jupyter team, Spawners, 2016, https://jupyterhub.readthedocs.io/en/stable/reference/spawners.html, last accessed 2020/02/28

[60] The JupyterHub Helm chart, https://github.com/jupyterhub/helm-chart, last accessed 2020/02/28

[61] The Kubernetes Authors, “Persistent Volumes”, 2020, https://kubernetes.io/docs/concepts/storage/persistent-volumes/, last accessed 2020/02/28

[62] NetApp Trident, 2019, https://netapp-trident.readthedocs.io/en/latest/introduction.html, last accessed 2020/02/23

[63] D. Batchelor and M. Satran, “Microsoft SMB Protocol and CIFS Protocol Overview”, https://docs.microsoft.com/en-us/windows/win32/fileio/microsoft-smb-protocol-and-cifs-protocol-overview, last accessed 2020/02/23

[64] Samba.org, “LinuxCIFS utils”, 2019, https://wiki.samba.org/index.php/LinuxCIFS_utils, last accessed 2020/02/23

[65] Docker Inc., “Runtime privilege and Linux capabilities”, https://docs.Docker.com/engine/reference/run/#runtime-privilege-and-linux-capabilities, last accessed 2020/02/23

[66] The Kubernetes Authors, “Pod Security Policies”, https://kubernetes.io/docs/concepts/policy/Pod-security-policy/, last accessed 2020/02/28

[67] A. Martin, “11 Ways (Not) to Get Hacked”, 2018, https://kubernetes.io/blog/2018/07/18/11-ways-not-to-get-hacked/, last accessed 2020/02/28

[68] Docker Hub, “Minimal Jupyter Notebook Stack”, https://hub.Docker.com/r/jupyter/minimal-notebook/, last accessed 2020/02/28

[69] Project Jupyter Contributors, “Setting up JupyterHub”, 2020, https://z2jh.jupyter.org/en/latest/setup-jupyterhub/setup-jupyterhub.html, last accessed 2020/02/28

[70] Docker Hub, “jupyter repositories”, https://hub.Docker.com/u/jupyter, last accessed 2020/02/28

[71] CKAN Association, “ckan”, https://ckan.org/, last accessed 2020/02/28

[72] UK Government, “Find open data”, https://data.gov.uk/, last accessed 2020/02/28

[73] Amazon Web Services, “Introduction to Amazon S3, https://docs.aws.amazon.com/en_en/AmazonS3/latest/dev/Introduction.html, last accessed 2020/02/28

[74] Minio, Inc, “minio”, https://min.io, last accessed 2020/02/28

[75] IEEE, “Interactive: The Top Programming Languages”, 2020, https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019, last accessed 2020/02/28

[76] pandas, https://pandas.pydata.org/, last accessed 2020/02/28

[77] SQLAlchemy authors and contributors, “The Python SQL Toolkit and Object Relational Mapper”, https://www.sqlalchemy.org/, last accessed 2020/02/28

[78] The PostgreSQL Global Development Group, PostgreSQL, https://www.postgresql.org/, last accessed 2020/02/28

[79] The Apache Software Foundation, “Apache Superset (incubating)”, https://superset.incubator.apache.org/, last accessed 2020/02/28

[80] J. Görner, “Beyond Jupyter Notebooks”, https://github.com/jgoerner/beyond-jupyter, last accessed 2020/02/28

[81] The Heapster contributors, “Heapster”, https://github.com/kubernetes-retired/heapster, last accessed 2020/02/28

[82] tribe29 GmbH, https://checkmk.com/, last accessed 2020/02/28

[83] Wikipedia contributors, “Reverse proxy”, Wikipedia, The Free Encyclopedia., 2020, https://en.wikipedia.org/w/index.php?title=Reverse_proxy&oldid=937552513, last accessed 2020/02/29

[84] Wikipedia contributors, “Transport Layer Security”, Wikipedia, The Free Encyclopedia., 2020, https://en.wikipedia.org/w/index.php?title=Transport_Layer_Security&oldid=943166788, last accessed 2020/02/29

[85] The Kubernetes Authors, “Using a Service to Expose Your App”, https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/, last accessed 2020/02/28

[86] The Kubernetes Authors, “Service”, https://kubernetes.io/docs/concepts/services-networking/service/#loadbalancer, last accessed 2020/02/28

[87] S. Dinesh, “Kubernetes NodePort vs LoadBalancer vs Ingress? When should I use what?”, https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0, last accessed 2020/02/28

[88] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus, “Knowledge Discovery in Databases: An Overview”, AI Magazine, 13(3), 57, 1992, DOI: 10.1609/aimag.v13i3.1011

[89] P. Chapman et al. "CRISP-DM 1.0: Step-by-step data mining guide”, Computer Science, 2000

[90] R. Jurney, “A manifesto for Agile data science”, 2017, https://www.oreilly.com/radar/a-manifesto-for-agile-data-science/, last accessed 2020/02/28

[91] The Apache Software Foundation, “Apache Airflow”, https://airflow.apache.org/, last accessed 2020/02/28

[92] The Kedro contributors, “Kedro”, https://github.com/quantumblacklabs/kedro, last accessed 2020/02/28

[93] A. Ronacher and contributors, Flask, https://palletsprojects.com/p/flask/, last accessed 2020/02/28

[94] https://www.hug.rest/, last accessed 2020/02/28

[95] A. Chilakapati, “Concept Drift and Model Decay in Machine Learning”, 2019, https://towardsdatascience.com/concept-drift-and-model-decay-in-machine-learning-a98a809ea8d4, last accessed 2020/02/28

[96] L. Hecht, “What the data says about Kubernetes deployment patterns”, 2018, https://thenewstack.io/data-says-kubernetes-deployment-patterns/, last accessed 2019/07/22

[97] Project Jupyter Contributors, “Zero to JupyterHub with Kubernetes”, 2019, https://zero-to-jupyterhub.readthedocs.io/en/latest/, last accessed 2019/07/22

[98] S. Conway, “Survey shows Kubernetes leading as orchestration Platform”, 2018, https://www.cncf.io/blog/2017/06/28/survey-shows-kubernetes-leading-orchestration-platform/, last accessed 2019/07/22

[99] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu, and W. Zhou, “A comparative study of Containers and Virtual Machines in Big Data environment”, arXiv:1807.01842v1

[100] S. Talari, “Why Kubernetes is a great choice for Data Scientists”, https://towardsdatascience.com/why-kubernetes-is-a-great-choice-for-data-scientists-e130603b9b2d, last accessed 2019/07/28

[101] Project Jupyter Contributors, “Customizing User Management”, https://zero-to-jupyterhub.readthedocs.io/en/latest/customizing/user-management.html, last accessed 2019/07/28

[102] OAuth community site, https://oauth.net/, last accessed 2020/02/23

[103] OpenID Foundation, 2019, https://openid.net, last accessed 2020/02/23

[104] S. Gerbel, “Implementation of a sustainable IT ecosystem for the use of clinical data to support patient-oriented research”, GI (Gesellschaft für Informatik): Medizininformatik (Medical Informatics), 2020, https://www.researchgate.net/publication/340925601_Implementation_of_a_sustainable_IT_ecosystem_for_the_use_of_clinical_data_to_support_patient-oriented_research, last accessed 2020/05/21