BIH/Charité Virtual Research Environment (VRE)

WHITE PAPER

Prepared by: Indoc Research  
Current Release: 1.2.0  
Last updated: May 15, 2021

# Overview

The BIH/Charité Virtual Research Environment (VRE) <https://vre.charite.de> is a data management platform that enables medical researchers to store, find, access, analyze and share their data, including sensitive data, promoting FAIR data principles and reducing barriers to biomedical research and innovation. The VRE is a module of the BIH Charite Health Data Platform systems (HDP), a component of the BIH Translational Hub Digital Medicine, and leverages that system’s generic big data infrastructure including its computing and storage resources. The VRE is developed in collaboration with Charité Business Division IT (GB IT) and industry partner Indoc Research.

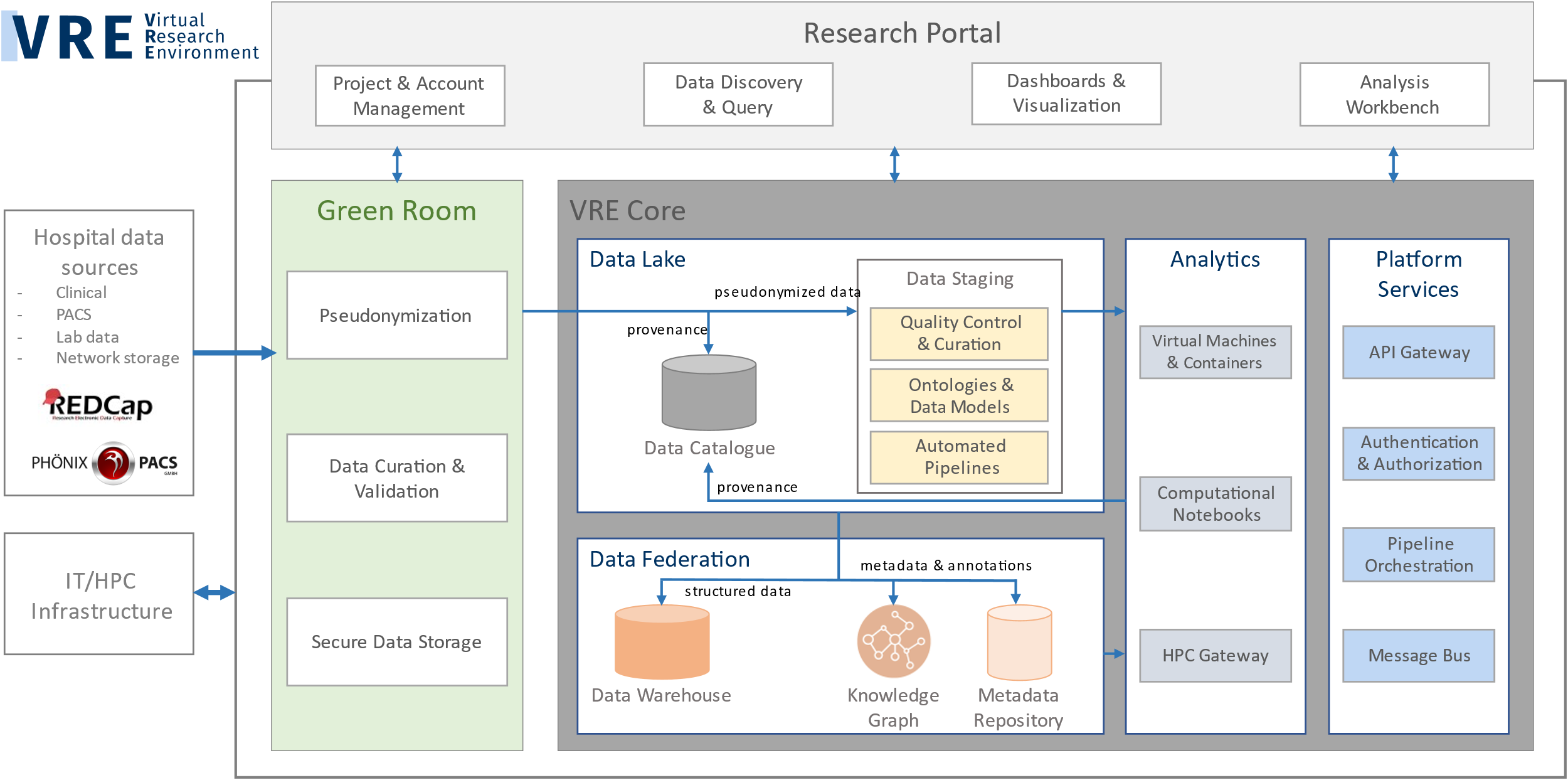


Figure . VRE high-level architecture

The current features of the VRE (illustrated in Figure 1) include: a web-based Research portal, a project-based and role-based access control and federated identity management system, a segregated Green Room zone to capture and pre-process sensitive data, analytics workbench tools for processing, analyzing, and visualizing datasets, automated ingestion of data from hospital-based data capture sources, a project-specific Data Warehouse for ingestion and query of structured datasets, a metadata repository and graph database for capture and query of metadata and lineage tracking, and support for ontologies and automated extraction and indexing of standard metadata fields to make data findable.

# Architecture and Implementation

## Summary

The deployment architecture of the VRE is described in Figure 2. The front-end functionality of the VRE is provided by the Research Portal, a web-based application that enables researchers and administrators to access and manage VRE resources and data. The back-end functionality of the VRE is deployed within the different architectural zones: Green Room zone, Utility zone, VRE Core zone, and VRE Extension, and consists of systems and services including identity management, access control, data operations, processing pipelines, databases, notification systems, and workbench tools.

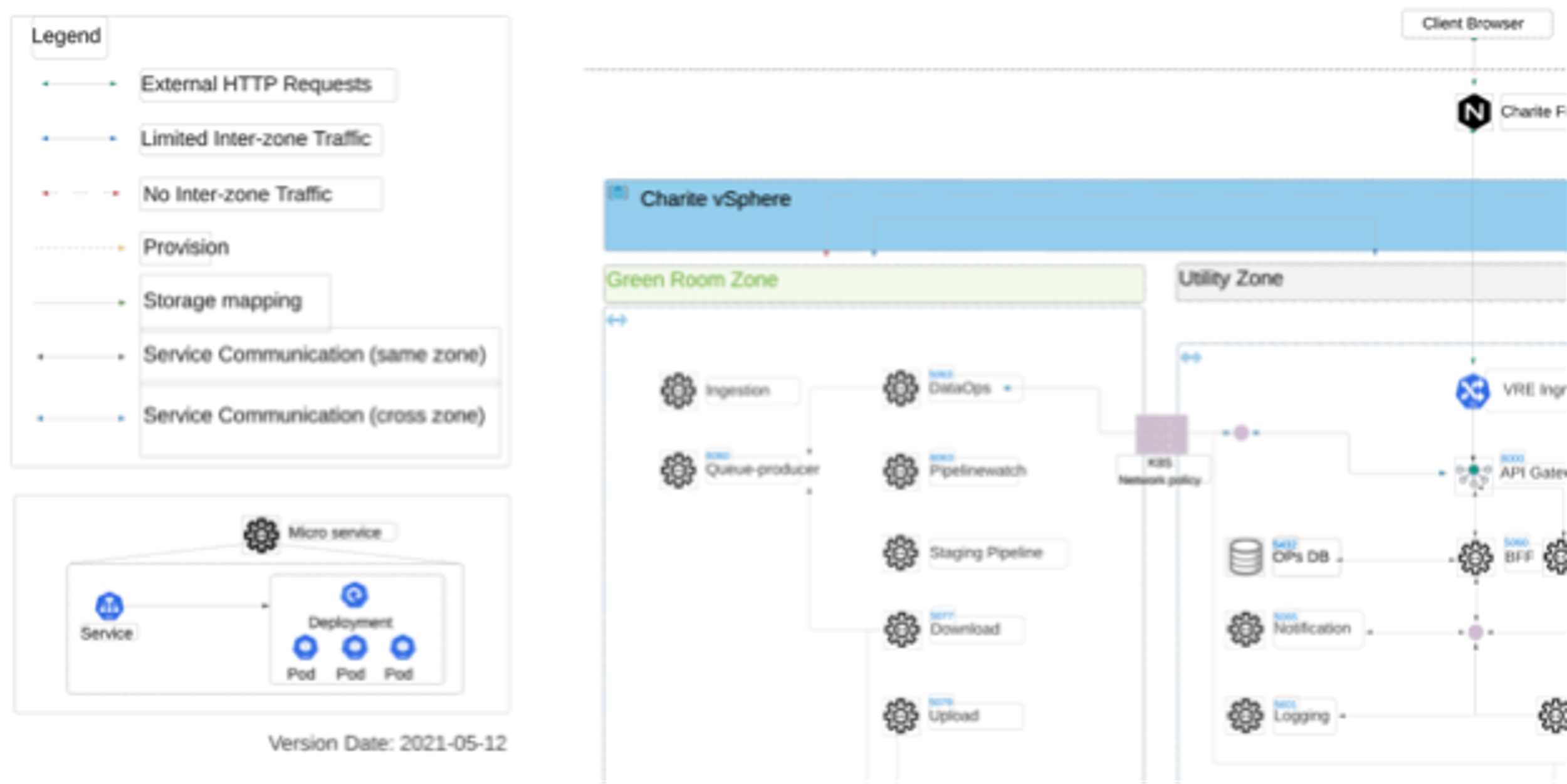


Figure 2. VRE deployment diagram

The VRE platform is hosted behind the Charité firewall. All user interactions with the data and platform content must take place via the research portal or VRE-managed command line or workbench utilities. The platform does not provide users with direct access or connection to the underlying resources such as storage, database, services, or VMs.

In alignment with Charité GB IT recommendation, the VRE platform is deployed on Virtual Machines (VM) within a flat network structure according to Charité's current research data handling practice. The isolation at the deployment layer uses the Kubernetes (K8S) namespace (see Section 2.2, Microservice Architecture), which separates these zones by restricting network traffic between the platform services at the application level. Only authorized access, defined by namespace network policies, is permitted from Green Room and VRE Core zones into the Utility zone. Direct service communication and data flow between Green Room and VRE Core zones are forbidden. In addition, the VMs assigned into each zone also restrict inbound and outbound traffic using IP tables.

The storage is provisioned into the VRE Core, Utility, and Green Room zones separately so that VRE Core services cannot access Green Room storage, but services running in the Green Room zone can access both Green Room storage and VRE Core storage (see Figure *2*Figure *2*. VRE deployment diagram and note that the Utility storage provisioning is omitted from the figure for simplicity). This is necessary because the Green Room services must be able to release the pseudonymized data to the VRE Core for further processing, analysis and sharing. In addition to physical storage separation, storage isolation is also achieved by exporting distinct NFS directories to the different zones.

As of VRE release 1.2.0, the VM consists of 27 VMs with 316 cores and 1840GB of RAM deployed to support the platform operations. Additional resources are planned to scale the platform’s capacity to support growing use by research projects.

## Microservice Architecture

The VRE adopts a microservice architectural pattern, typically consisting of a collection of loosely coupled but highly cohesive self-contained services that communicate through either REST APIs or message brokers. Each of these services has a single responsibility and does not carry many business dependencies to other services. Therefore, microservices can be developed, tested, deployed, upgraded, and maintained independently, allowing for greater development and deployment agility. At the same time, each service is often stateless and therefore can be executed and scaled individually to ensure resilience and scalability due to advancements in containerization technologies.

### Containerization

Containers provide a sandboxed and resource-limiting running environment for applications by packing the build of the software together with the dependencies, such as Python libraries, in portable images. In comparison with the more conventional Hypervisor-based virtualization, i.e., Virtual Machines, container-based virtualization only isolates the system processes and therefore provides a more light-weight environment that delivers fast performance and small boot-up overhead. Among the multiple OCI-compliant container runtimes, Docker is one of the most mature and widely adopted formats, hence the VRE platform implements Docker for service containerization.

An image manifest (“Dockerfile”) is needed to build a software service into a Docker image. Figure 3 shows an example of Dockerfile used by the VRE queue producer service. This Dockerfile describes the instructions for the build by defining the source image to be included and outlining the commands to be executed: create user, copy and build the source code, install the dependencies, and finally, run the bootstrap scripts to bring up the service after the container starts.

Once the service is successfully packaged into the Docker image, one can spin up the container instances from that image on any Linux host that has a Docker runtime. Each of these microservice containers typically exposes a HTTP port with one or more associated API endpoints that in turn can be consumed by other platform services.

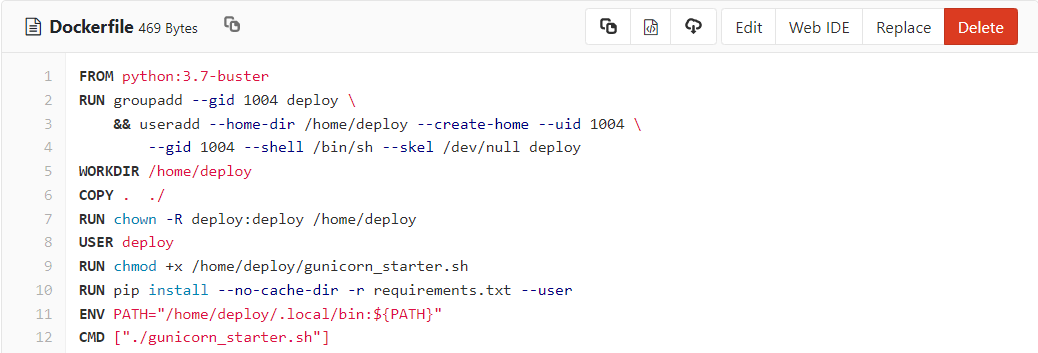


Figure . A sample Dockerfile for VRE queue producer service

### Service Orchestration

Kubernetes is a natural platform for microservices as it can handle the orchestration required to deploy multiple instances of multiple individual microservices. Charité IT does not currently have a managed orchestration framework such as OpenShift so the VRE Kubernetes cluster was developed from the scratch to manage the deployment of the service containers on a group of VMs.

The smallest deployment unit in a Kubernetes cluster, called a pod, may include one single service container with or without the sidecar containers for logging or monitoring purpose. Kubernetes uses the Deployment YAML file to create the pod and schedule the pod onto the appropriate cluster node for execution. An example pod status is shown in Figure 4. This YAML file declares the desired running state of the pod, for instance, the name of the service container, the number of replica and HTTP ports. Kubernetes controller will then bring the state of cluster node to meet the requirements.

Table

Description automatically generated

Figure . A snapshot of the pod status in VRE Green Room

Horizontal scaling of a microservice is made easy in a Kubernetes cluster. Figure 5 depicts an example YAML deployment file from the VRE Green Room: if the value of replicas (line 9) is set to a number greater than 1, Kubernetes will create multiple pods for the VRE queue producer service to match the number of replicas and bundle them behind a load balancer to share the load. The download service in Figure 4Figure 4. A snapshot of the pod status in VRE Green Room is an example in which the VRE attempted to run four instances of the download services concurrently. In addition, Kubernetes also automatically handles service scaling by tracking the system metrics such as CPU or RAM usage.

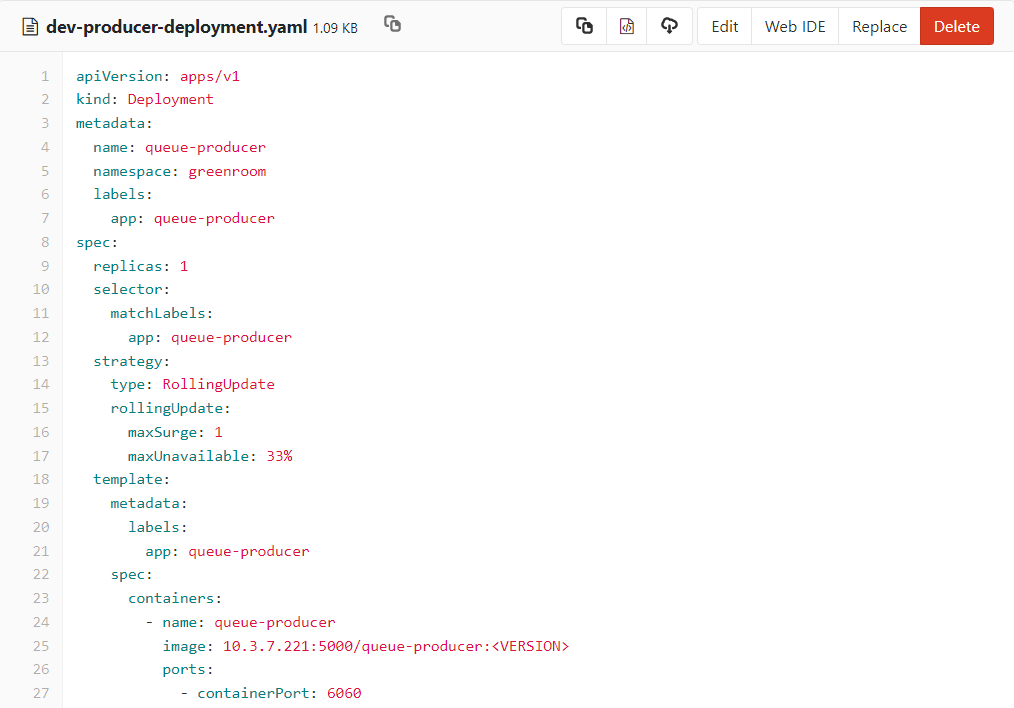


Figure . Deployment YAML file of VRE Green Room queue producer service

Kubernetes allows the creation of virtual clusters inside a deployment, known as namespace, to logically isolate the pods and other cluster resources such as cluster nodes and storage volumes. It is a good common practice to use a namespace to establish security boundaries through network policies such that sensitive workloads always run on a dedicated set of cluster nodes. As illustrated in Figure *2*, the VRE deployment uses the Kubernetes namespace to divide the cluster into Green Room, Core, Utility, and Workbench zones. The network policies enforce the traffic between the zones; for instance, no services running in the Core zone can directly visit the services and resources hosted in the Green Room. This approach reduces the risk of a sensitive service or data volume being accessed through a less sensitive service.

### Deployment Cycle

The microservices pattern is also known as an agile architecture and therefore shares a development culture with DevOps, where continuous integration (CI) and continuous delivery (CD) can make quick and frequent production releases. Kubernetes further enables the automation of building, deploying, and configuring infrastructure resources using the declarative deployment files that live in the same repository where the source code is hosted.

The VRE development team adopted the common 3-tier deployment model, known as DSP, that consists of development, staging, and production environments. Developers first develop the service and complete the templates for both the Dockerfile and Kubernetes deployment YAML file mentioned in previous sections, then push the code to the dev branch in the git repository. The CI/CD pipelines, defined by a Jenkinsfile, are automatically triggered by the git commit to copy the repository to a Jenkins runner where the following tasks are executed to complete the deployment on the dev environment: (1) The Jenkins pipeline builds the source code of the service into a Docker image and execute unit tests, (2) If tests are all passed, the image is pushed to a private container image registry used by Kubernetes, (3) Kubernetes will pull the image and deploy it onto the cluster nodes according to the instructions provided in the deployment YAML file.

Once developers are satisfied that the service can go to staging for user acceptance testing (UAT), they commit the source code together with Kubernetes configuration files to the staging branch and the similar CI/CD steps repeat. If issues are found in staging deployment, developers will fix the issues in their dev branch and try to commit again. Unless the fix is minor, no actual code development should directly take place on staging servers. The service will finally be pushed into the production environment if the staging deployment is thoroughly tested and approved by the UAT team.

## VRE Portal

Most researcher interactions with the platform take place using the Research Portal. Other specialized data interactions may take place using the command line tool or VRE Extension, described in subsequent sections.

The VRE Portal provides functionality for user authentication and authorization, project- and role-based upload, access and processing of data, administrative features including user management and project configuration, and access to workbench tools and project Wiki.

The VRE Portal keeps track of user interactions such as upload, download, and creation of user accounts to ensure that all interactions and operations are authorized and are traceable. Users are not able to directly interact with back-end services or access Green Room data directly (see Section 2.4, VRE Green Room).

The research portal front-end uses React, an open-source JavaScript library, to describe the user interface (UI) and graphical presentation of the data models returned from the back-end services. In production mode, the portal front-end is compiled into a static build package by the React build tool for easy deployment and fast page loading. The static build of the portal front-end is placed behind the VRE Ingress Controller, which is behind Charité’s proxy and firewall. The portal front-end communicates with back-end services through an internal VRE API Gateway which authenticates the requests before routing them to the appropriate back-end services. The details of the back-end services and zones are described in the following sections.

## VRE Green Room

The Green Room is a restricted zone isolated from other VRE zones. Since research data uploaded to the VRE may include sensitive personal identifiers, the Green Room provides a secure environment in which these data can initially land and undergo review, validation, and, if necessary, transformation to remove or limit sensitive information before the data are made available to other members of the project team for scientific analysis in the VRE Core.

Data landing in the Green Room may be ingested from Charité data sources using a direct ingestion pipeline, e.g., a research container in the Charité Phönix PACS, or from file upload by researchers. Dedicated storage allocated to the Green Room is isolated from other VRE zones, ensuring that Green Room data can only be accessed by authorized services deployed within the Green Room itself, and not by services deployed in other zones or in other Charité IT environments.

The Green Room zone is deployed as a Kubernetes namespace which hosts several data operation services to support upload, download, and annotation inside the Green Room. Staging pipelines and other event-sourcing auxiliary services are also deployed inside the Green Room to streamline pseudonymization processes. Network traffic between the Green Room and other VRE zones is strictly controlled such that only a few services in the Utility zone are authorized to directly communicate with Green Room services. The DataOps service can access both Green Room and VRE Core storage to facilitate copying approved data from the Green Room to VRE Core; the remaining Green Room services can only access Green Room storage. Green Room services include the following:

*Upload Service:* receives user-uploaded data in small chunks, combines them, and places the complete data file in the appropriate project folder. Once the data are saved, this service notifies the Entityinfo Service in the Utility Zone to record the upload event by capturing the file name, user account, storage location, and timestamp. The Upload Service also emits a message to the Message Bus to indicate completion of the upload, which can be acted upon by the Message Listener Service to trigger downstream processes such as pipelines. The Upload Service cannot be invoked by any other services outside of the Green Room.

*Download Service:* receives data download requests, compresses the requested data into a single archive, and generates a one-time download link that is shared with the requesting user. This link does not reveal the actual file storage path and is set to expire 5 minutes after it is generated.

*Queue Producer:* receives Green Room events (such as uploads and downloads) and pushes these events and their associated information into the Message Bus, which later are consumed by the Message Listener Service for asynchronous processing.

*DataOps Service:* performs all Green Room data operations on files stored in Green Room storage, other than upload and download. This includes annotating, tagging, and organizing files into collections.

*Pipeline Watch:* monitors Kubernetes job executions and passes the latest job status to the Entityinfo Service in the Utility zone.

*Staging Pipeline(s):* deployed optionally on a project-specific basis at the request of the Project Administrator to perform automated filtering or masking of data fields that contain sensitive personal information such as direct identifiers.

## VRE Core Zone

The VRE Core Zone stores data that have been received from Green Room storage, signifying that these data are prepared and ready for analysis by members of the research team. The VRE Core zone is designed to support a wide range of research data operations and includes other components such as a data warehouse.



Like the Green Room, the VRE Core zone is deployed as a Kubernetes namespace, hosting a several data operation services and the platform-wise Data Warehouse. No direct network traffic is allowed between the VRE Core and the Green Room. Only the VRE Core storage is accessible by VRE Core services. VRE Core zone services including the following:

*DataOps Service:* similar in function to its Green Room counterpart, the VRE Core version of the DataOps service can only access the data stored in the VRE Core Zone. In addition to data operations such as annotating, tagging, and organizing files into collections, the VRE Core DataOps Service also extracts, loads, and transforms data flowing in and out of the Data Warehouse.

*Data Warehouse:* aggregates data points derived from transactional data or metadata extracted from research datasets. Each research project on the VRE is assigned its own Data Warehouse instance. Within each instance, database schema and permissions are used for fine-grained access control, ensuring that project members only access the data within their own Data Warehouse instance.

*Upload Service:* accepts data upload initiated by users within the VRE workbench.

*Download Service:* downloads the data from the VRE Core storage.

*VRE Core Extension:* a sub-zone of the VRE Core zone that contains specialized deployments of third-party tools, enabling the VRE to accommodate a variety of external applications that require custom configuration and technical resources not available through the VRE Portal.

## VRE Utility Zone

The Utility Zone, deployed as another Kubernetes namespace, contains shared back-end services to support the operation of VRE systems, and exchanges and routes requests between the Green Room and the VRE Core. No research data are stored in Utility Zone, and the Utility zone cannot access either the Green Room or VRE Core storage. VRE Utility zone services including the following:

*Ingress Controller:* all services, including the API Gateway, are inaccessible from outside of the VRE deployment. The Ingress Controller defines rules for external connectivity, e.g., from the Charité out-facing proxy, to the VRE deployment, including URL rewriting, upstream services, etc.

*API Gateway:* a single-entry point situated between the Ingress Controller and VRE platform services. The API Gateway is responsible for request routing, composition, rate limitation, and monitoring. It handles requests by routing them to the appropriate back-end services. If there are failures in the back-end services, the API Gateway can return cached or default data. In the VRE deployment, the API Gateway is also connected to the internal Identity Provider (IdP) to protect the back-end APIs from unauthorized access.

*Back-end for Front-end (BFF):* a service dispatcher that further manages the routing of requests to back-end services and aggregates the results from multiple downstream calls before returning the final response to clients. This architecture also makes it possible to centralize fine-grained authorization control. In this design, BFF also acts as a bridge to facilitate and control data exchange between the Green Room, Utility Zone, and VRE Core.

*Entityinfo Service:* provides metadata management and governance capabilities to build data catalogues and govern data assets stored on VRE platform. Custom metadata types can be defined to track metadata associated with user data saved on VRE storage and databases. These custom metadata types allow the details of data operations (such as upload, download, pipeline executions) and relationships (yields, modifies) between the data to be captured as indexed entities that are searchable via APIs.

*Ops Database:* the Ops Database in the Utility Zone does not store any research data but provides operational support for the VRE platform. Most data tracked by the Ops Database are transient or temporary and subject to expiry, for instance, upload / download status, project configuration, and user invitation history.

*Graph Database:* similar to the Ops Database, the Graph Database (Neo4J) provides operational support to the VRE portal. It keeps track of projects, relationships between projects, and user memberships created on the VRE portal. The Graph Database brings flexibility in modeling the complex hierarchy of the organization and its research initiatives.

*Kubernetes Master Nodes:* three master nodes are deployed in the Utility Zone to manage and orchestrate the Kubernetes worker nodes used by the VRE system. For high availability, each master node runs on an individual virtual machine. Each master node includes an API server that communicates with external commands, a manager that controls the worker node cluster, a database that tracks the cluster state, and a scheduler that schedules a pod execution on a worker node according to the resource plan.

*Notification Service:* a back-end service that is responsible for outgoing VRE platform notifications such as emails. The current implementation uses an SMTP server provided by Charité IT.

*User Account Management Service:* integrates Keycloak as the IdP to provide the authentication and user account management service to the VRE. Keycloak is a widely adopted open-source software tool that follows standard security protocols to provide identity management and single sign-on solution. Keycloak acts as a proxy to retrieve VRE user profiles (username, email address) from Charité’s Active Directory for the purposes of provisioning user accounts after users provide their consent to the platform Terms of Use.

*Logging Service:* VRE platform services follow the convention of either writing their logs to standard output or error streams (i.e., stdout and stderr) or, where containerization is used, employing a sidecar pattern in which a service is accompanied by another dedicated logging service. In the latter case, logs produced by a service are then picked up by a local agent that ships the log to the central log management system, such as Elasticsearch, for further analysis.

*Private Registry Server:* the internal registry server that hosts container images of VRE services and configurations. The Kubernetes cluster pulls containerized services from this server prior to scheduling these services on worker nodes. This setup provides a protected container repository so that container images can only be shared within the VRE platform.

*CI/CD Runner:* a collection of scripts that automatically build and configure VRE source code into images that in turn are pushed into the Private Registry Server.

## Workbench

The VRE Workbench provides users with a personalized analysis environment and analytics tools to process and visualize their data. Workbenches are deployed on a project basis using a Kubernetes namespace to provide isolation between projects. Each VRE project is allocated computational resources (CPU cores, RAM, storage) to run Workbench services, and these resources can be scaled based on demand. Currently, only data stored the in VRE Core are provisioned into the Workbench (in read-only mode). At the time of writing, the following workbench tools are deployed in the VRE:

*JupyterHub:* users can access the interactive computational notebooks based on the JupyterHub multi-user server to upload, edit, and execute scripts (coded in Python, R, or other languages).

*Apache Guacamole RDS:* a remote desktop service (RDS) based on open-source Apache Guacamole software that allows users to directly access virtual desktops (e.g., a Linux or Windows operating system) and shell terminals to launch a wide range of applications including graphical interfaces and a container engine. When permissions allow, users may download, install, and import standard or containerized Linux-based software within this environment.

*Apache Superset:* an open-source data visualization and data exploration application that enables users to easily query, analyze, and visualize structured data.

## Command Line Interface

Users can access the VRE command line interface (CLI) within Workbench workstations configured through Apache Guacamole RDS. The CLI allows VRE users to perform common data operations without having to use the web portal. For instance, users can copy data stored in the VRE Core into their Workbench local storage, use Workbench tools to perform analyses on these data, and upload analysis outputs back into VRE Core storage. Users can also look up their project’s attribute templates to perform advanced file annotation and query the VRE Core data based on tags.

## XWiki

XWiki is an advanced open-source platform deployed in the VRE to support creation and sharing of Wiki document content within a research project, and to provide shareable digital content to public visitors of the VRE. Project Wikis are deployed using XWiki’s (sub)wiki feature to restrict content based on project membership. The publicly available Wiki contains a Developer Guide, Platform Administrator Guide, and VRE User Guide (see Section 4, Community Engagement . The public Wiki is accessible to members of the public without the need to login to the VRE.

## Authentication and Authorization

VRE user accounts are federated with the Charité Active Directory (AD). Invitations for Research project members to join the VRE are initiated by a Project Administrator or a Platform Administrator for projects approved by the Use and Access Committee. Research members who are not employees of the Charité must additionally complete an application form and provide a personal identification (e.g., last six digits of a passport number) as an additional measure of identity verification in order to be authorized for inclusion in the Charité central AD.

User accounts in third-party applications provisioned through the VRE Portal (e.g., JupyterHub, XWiki) are authenticated by Single Sign-On (SSO) with Keycloak, the VRE identity provider. The built-in LDAP/AD provider of Keycloak is used to federate the user and group records in the “vre” Organization Unit (OU) managed by Charité AD. The VRE Keycloak service does not store the user information; instead, it authenticates the user-provided credential against the Charité AD.

The VRE service components - the front-end, API Gateway that connects all back-end services, and workbench tools - are registered into Keycloak as individual clients using OpenID Connect (OIDC) as the authentication protocol. The typical authentication workflow in VRE is described as the follows: VRE portal asks the Keycloak to authenticate a user. After a successful login, VRE portal receives an access token that contains username, email, other profile information, and access details such as role mapping. The access token is digitally signed by Keycloak and can be used by other registered clients to invoke other services on behalf of the user. The service that receives the request then extracts the access token, verifies the signature of the token, and decides based on access information within the token whether to process the request.

## Integration of BIH HPC Cluster

The BIH High Performance Computing (HPC) cluster is fully managed and maintained by BIH IT. A System Administrator mounts project folders from VRE Core storage in read-only mode on cluster nodes; these folders are available only to project members. With the AD federation in place, VRE users can log into a job submission node by IP address via Workbench tools integrated into the VRE Portal.

The HPC environment provides a scratch space for users to store intermediate results generated during HPC computations. Although the scratch space is a shared network storage area, users have private scratch folders within this space that are only accessible to themselves, thus ensuring segregation of user project data. The scratch space is purged periodically in accordance with data retention policies of the BIH HPC facility. Users are instructed to store final outputs of their HPC jobs within their home folder, which can then be pushed back to VRE Core storage using the VRE command line interface

## Integration of Hospital Data Sources

Data from Charité research and clinical systems can be copied into the VRE to allow researchers to aggregate multi-modal research datasets in a common environment, and process these data using VRE analysis tools. Data source being integrated with the VRE include (1) the Phönix picture archiving and communication system (PACS) which stores radiologic image files such as MRI and fMRI, as well as other physiological data types such as EEG and MEG; (2) the REDCap electronic data capture and clinical data management system.

### PACS Integration

Researchers use the Charité Phönix PACS to store, manage and access imaging and other data. Typically, research analysis of such data requires users to query and extract data manually from the PACS into their local data storage environment which may be limited by capacity, computing power, and appropriate sharing controls. The VRE integration with the PACS permits secure data extraction with minimal user intervention to make these data available to project members for processing using the VRE Portal resources and workbenches.

Automated integration is managed on a project-basis. The Charité PACS administration group creates a project-specific PACS container, a logically isolated storage space within the Phönix system that will hold the datasets of interest, and configures rules defined by the Project Administrator to automatically copy datasets into this container. The VRE PACS Data Pipeline then periodically extracts the most recent project data from the PACS container into the VRE Green Room.

### REDCap Integration

Charité researchers utilize one or more centralized production instances of REDCap deployed within the Charité HDP to collect and manage structured, form-based data for their research studies. Researchers may wish to access these data within the VRE in order to integrate them with other modalities and to conduct data analyses. To accomplish this, the Charité Health Data Platform administration group first onboards the REDCap data into the Charité Hadoop-based data infrastructure and maintains an automated data extraction pipeline that queries the given REDCap project and extracts the required data into an Apache Hive data warehouse, from which, the VRE REDCap Data Pipeline pulls data into a VRE project-specific data warehouse. Researchers then can query their REDCap data using the Apache Superset data exploration application from within their project workbench.

## Data Storage

VRE data are stored on central virtual servers and storage infrastructure provided by the Charité IT division (GB-IT). Data storage infrastructure used by the VRE includes HPE 3PAR SAN storage allocated to the VRE VMs managed by VMWare, and Qumulo storage exported to VMs over NFS. Disk level encryption is active on both HPE and Qumulo infrastructure, providing a measure of security that protects VRE data when disks are powered off or physically removed. Furthermore, access to GB-IT premises is subject to the Charité central technical and organizational measures. Appropriate measures to control access are centrally implemented and are regularly monitored. Access to the server rooms is only permitted to registered persons. Access is logged by the security service.

VRE project data and VM images are regularly backed up based on standard Charité GB IT backup systems and policies. Backup and restoration measures are further described in the Operating Concept document

# Policies and Compliance

Data controllers have a duty to protect the privacy of personal health information of the research data subjects. Even when direct identifiers such as names or faces are removed, biomedical data contains additional features such as brain images or genetic material that are unique to an individual and could lead to reidentification if data processing is not carefully controlled. The design and development of the VRE incorporates principles of “privacy by design and by default” (GDPR Article 25) to offer researchers an environment to process and analyse research data in compliance with their obligations as data controllers. The platform is hosted within the infrastructure of the BIH-Charite IT Health Data Platform and is governed by strict Technical and Organizational measures appropriate to the collection, storage, and processing of personal health data. The VRE is in the final stages of an independent legal evaluation to confirm that the security and privacy measures support the processing of personal health data for research purposes in accordance with GDPR. The appropriate policies and procedures have been developed, including specific data protection requirements, technical and organisational measures and risk assessments. There are clear definitions of roles, rights and responsibilities for all categories of users, relating these to the GDPR description of Data Processors and Data Controllers.

Access to the platform is governed by transparent Use and Access policies that promote fair and democratic access to the VRE. Researchers wishing to use the VRE to process research data may apply for access and formalize their intended usage of the platform with concrete agreements including a Service Agreement, Data Processing Agreement, Terms of Use, and evidence of a positive Research Ethics Board evaluation. For data that contain biomedical information of natural persons, researchers must also prepare a data protection impact assessment that has been reviewed by the relevant Data Protection authorities. User accounts are granted with project-based and role-based controls that allow different types of users to interact with the platform and data holdings according to the principle of access minimization.

# Community Engagement

The VRE is a co-development project of the Charite IT Division (GB-IT) and contract partner Indoc Research (Canada). Development of the platform started in April 2020 as a community-driven, iterative effort in which the input of stakeholders (researchers, IT, management, data protection officials) informs the ongoing development of the features, front-end interface, and backend architecture.

Within the development period of 12 months, Indoc Research coordinated weekly meetings open to stakeholders from BIH/Charite and external organizations interested in the VRE. In these sessions the development team updated the community on the progress of the VRE and collected feedback on features and new requirements. Stakeholder and participants presented research use-cases and formed new collaborations and synergies between groups. In the final month of the first development phase (April 2020 to May 2021), the VRE development project has thus engaged a community of over 151 individuals from 24 institutions in seven countries across Europe and North America. Ongoing user acceptance testing by researchers from several projects has been positive and is shaping the future development priorities of the platform.

The VRE supports FAIR data principles in the developer community by publishing and updating open-source development code and documentation in the VRE Wiki (see <https://vre.charite.de/xwiki/wiki/vrepublic/view/Main/>) and by deploying and supporting the use of open-source workbench tools that are commonly used among the research community (Jupyter notebooks, Python libraries, Docker containers, Apache Guacamole remote desktop gateway, Apache Superset and XWiki).