digipolis VERLAG

Whitepaper #4 I Juni 2025

Building the IKID Sandbox:

A Deep Dive into the Architecture and Implementation of a Platform for multi-dimensional AI Education

ai.hdm-stuttgart.de/research/ikid

- **Malte Grosse**
- **Cornelius Specht**
- **Peter Thies**

INTEGRIERTE KI-LEHRE

HOCHSCHULE DER MEDIEN

IAAI Institute for Applied Artificial Intelligence

IDE HOCHSCHULE DER MEDIEN

Baden-Württemberg MINISTERIUM FÜR WISSENSCHAFT, FORSCHUNG UND KUNST

GEFÖRDERT VOM Bundesministerium für Bildung und Forschung

INTEGRIERTE
KI-LEHRE

Whitepaper #4: Building the IKID Sandbox

A Deep Dive into the Architecture and Implementation of a Platform for multi-dimensional AI Education

by Malte Grosse, Cornelius Specht & Prof. Dr. Peter Thies

Funding organisation:
Bundesministerium für Bildung und Forschung (BMBF) and *Ministerium für Wissenschaft, Forschung und Kunst (MWK)* des Landes Baden-Württemberg (Förderkennzeichen: 16DHBKI040)

IKID Project Management:
Prof. Dr. David Klotz – klotzd@hdm-stuttgart.de
Hochschule der Medien Stuttgart, Nobelstraße 10, 70569 Stuttgart

Editor:
Marcel Schlegel

Design and Layout:
Michelle Stegner and Marcel Schlegel

Cover image:
Graphic generated with *Dall:e*

Publisher:
Digipolis Verlag
Sina Klauke, Tramweg 8, 77966 Kappel-Grafenhausen
kontakt@digipolis-verlag.de
www.digipolis-verlag.de

https://doi.org/10.70481/rck1-6tm3

Suggested citation for this whitepaper edition:
Grosse, Malte/Specht, Cornelius/Thies, Peter (2025): Building the IKID Sandbox: A Deep Dive into the Architecture and Implementation of a Platform for multi-dimensional AI Education. In: Whitepaper-Serie zum Forschungsprojekt IKID: Interdisziplinäres KI-Exploratorium. Kappel-Grafenhausen: Digipolis Verlag. https://doi.org/10.70481/rck1-6tm3

Table of Contents

Building the IKID Sandbox

Abstract

The IKID Sandbox is a platform designed to facilitate the research and application of artificial intelligence (AI). It provides a secure environment for users of all levels of experience in the field of AI, with a particular focus on security, efficiency and user-friendliness. The modular architecture using Kubernetes makes it possible to ensure the availability and scalability of the chosen open-source applications. Role-based access control and isolated training environments protect sensitive information. The sandbox provides tools for various user groups, including a powerful web-based development environment, a data pool for data sharing and reliable AI training pipelines for computationally intensive machine learning tasks.

This white paper is generally aimed at people who want to gain insights into the architecture of the Sandbox. Lecturers can also use this white paper to familiarise themselves with the tools provided and integrate them into their syllabus. For students, it offers the opportunity to explore the used containerised software components or the general software architecture. ■

Disclaimer: This document was originally composed by human authors. To enhance clarity and readability, it has been further refined using Large Language Model (LLM) technology, demonstrating the capabilities of modern AI tools in augmenting human creativity and productivity.

01 Introduction

Interdisciplinary AI Exploratorium: Integrated Teaching for the Responsible Use of Artificial Intelligence based on Physical-Virtual Demonstrators is a project funded by the *Federal Ministry of Education and Research* and the *Ministry of Science, Research and Arts Baden-Württemberg*: **I**nterdisziplinäres ***KI****-Exploratorium: Integrierte Lehre zur verantwortungsvollen Nutzung Künstlicher Intelligenz auf Basis physisch-virtueller* ***D****emonstratoren* or short **IKID.**

The majority of AI teaching courses often present a narrow perspective, focusing solely on the technical aspects of AI. However, responsible AI usage demands a comprehensive understanding of its impact on society, which includes economic, legal, and ethical considerations. The IKID project aims to bridge this gap in higher education by adopting an interdisciplinary approach.

The project introduces the concept of an "AI Exploratorium" ("German: ‚KI-Exploratorium"), a unique environment that showcases the complexity of AI through different interactive use cases, such as face recognition. To ensure accessibility and accommodate diverse student backgrounds, a user-friendly IT infrastructure is essential. Therefore, next to other initiatives within the AI exploratory, the project has opted for a containerized server infrastructure, allowing students to explore and develop AI use cases using their own devices or through a shared platform: The **IKID Sandbox.**

This infrastructure serves as the foundation for an integrated teaching environment, enabling concurrent student projects and research activities. The Sandbox offers a collaborative space where lecturers and students can work independently on case studies within a shared data pool. Additionally, the Sandbox playground enhances machine learning, including advanced training pipelines, providing a safe and controlled environment for students to experiment and gain valuable insights into AI usage.

The following document outlines the Sandbox's core requirements: a browser-based development environment, a use case environment for collaborative AI development, a robust training environment for complex model training, and a data pool with diverse datasets. It then delves into the technical architecture, detailing the hardware specifications and a three-part virtual machine setup for efficient resource management. The whitepaper describes the implementation of each software component, including containerization, ingress and reverse proxy for secure communication, authentication and authorization with, a centralized database, a Git-based version control system, a package repository for software artifacts, and an object storage solution. The paper concludes by highlighting the Sandbox's flexibility, scalability, and user-friendliness, emphasizing its potential as a valuable resource for AI education and its ongoing evolution to integrate emerging AI technology. ■

02 Requirements

General

The development of the Sandbox was a collaborative effort involving lecturers and students to ensure it met their diverse needs. The IKID Sandbox caters to a diverse range of users with varying levels of technical expertise. **Basic users**, primarily focused on the societal impacts of AI, such as those teaching or exploring use cases in ethics, business, and law, can utilize the Sandbox's pre-built interactive examples, demos, and course material. They can also leverage the platform for collaborative assignment editing and feedback. **Advanced users**, typically from IT backgrounds, are empowered to develop and share their own examples and demos, leveraging the Sandbox's powerful infrastructure for tasks like model training and deployment. This dual approach ensures the platform's relevance and accessibility for both technical and non-technical audiences engaged in AI education.

Three key areas emerged from discussions with these stakeholders that the Sandbox should address:

Firstly, a platform-independent development environment was deemed essential because it enables users to access the Sandbox remotely, ensuring that learning could continue uninterrupted, regardless of physical location. Such a development environment also allows for a more flexible and dynamic learning experience, catering to diverse user preferences and capabilities.

Secondly, a training environment for AI models was identified as a crucial component. This space would allow students and lecturers to experiment, train, and test their AI models in a controlled environment. It provides a safe space to explore the potential of AI, gain practical experience, and develop valuable skills in this rapidly evolving field.

Lastly, the need for a data pool containing case studies and training data was emphasized. This resource would offer real-world examples and experimental data, enhancing the teaching and learning of AI following the principles of research-based learning. By providing access to diverse datasets, students can apply their knowledge to

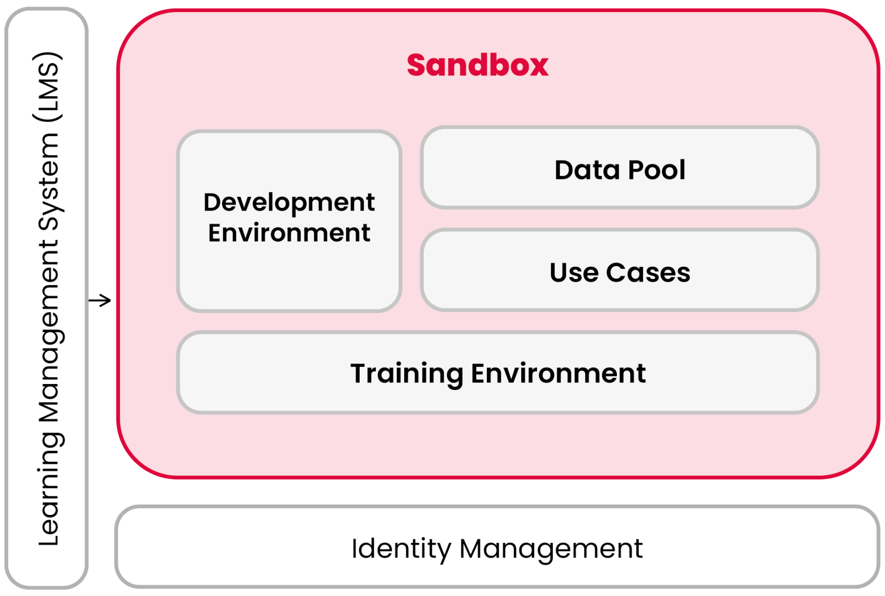

Figure 1: Overview of the Sandbox Components.

practical scenarios, developing critical thinking skills and a deeper understanding of AI applications.
These components collectively form a comprehensive Sandbox, catering to the needs of lecturers and students in the field of AI education. It provides a versatile and dynamic learning environment, fostering experimentation, remote accessibility, and a practical understanding of AI and its potential applications.

Based on the identified requirements, the Sandbox's conceptual design should include four major functionalities:

Firstly, an interactive, browser-based, (i) **Development Environment** provides beginners and advanced users with an isolated and safe AI playground.

Second, the platform features a web-based, interactive (ii) **Use Case Environment** designed to provide scalable and efficient computational resources. Users can create, edit and run use cases in their browsers, benefiting from the integrated environment to dynamically allocate computing power, manage resources, and ensure high availability. This setup allows for seamless collaboration, reproducibility, and the ability to handle complex data science workflows and provide a playground for AI, making it ideal for researchers, educators, and developers.

Third, the (iii) **Training Environment** empowers users with a suite of flexible and advanced toolsets. This includes access to Graphics Processing Unit (GPU)-supported high-memory/Central Processing Unit (CPU) instances, ideal for tackling computationally intensive tasks and long-running training pipelines. For enhanced reproducibility and collaboration, the environment utilizes isolated containers with full version control, all seamlessly managed by a robust infrastructure. This ensures a consistent and scalable platform for scientific workflows.

Fourth, the (iv) **Data Pool** provides for a wide range of users by offering a variety of flexible data storage options. Basic and advanced users alike can leverage the user-friendly Sandbox browser-based explorer for intuitive data management. Figure 1 depicts the IKID Sandbox environment. The existing Learning Management System and Identity Management are external services which are provided by the educational institution.

Hardware Components

Delineating from the Sandbox architecture and intended functionalities, the requirements for the server hardware were diverse and demanding, needing to balance powerful GPU computing capabilities for AI model training with the ability to handle intensive CPU application workload. This led to the following specifications: To meet these demands, the server was equipped with an *AMD* 2x 32 Core CPU, ensuring enough processing power for running development environments and database applications. The 8x 64GB memory modules provide robust support for handling large datasets and complex tasks. For storage, a combination of SSDs and HDDs has been installed: 2x 1.6TB Fast SSDs offer quick access to frequently used data, while 6x 0.48TB SSDs provide additional high-speed storage. Additional 8x 2.4TB HDDs offer space for larger datasets and long-term storage needs. Additionally, an *NVIDIA* A100 80GB GPU has been included, ensuring the server has the capacity for AI model training and handling demanding computational tasks. The server also employs additional hard disks as part of a comprehensive backup solution, ensuring data redundancy and security. This server configuration balances performance and storage, catering to the diverse needs of AI model development and traditional application demands. It provides a versatile platform capable of meeting the requirements specified by the stakeholders. ■

03
Design and Implementation

Overview of the IKID Sandbox Architecture

From a technical perspective, the primary objective was to devise an architectural framework for the IKID Sandbox that embodies stability, safety, flexibility, and extensibility. To achieve this, distinct application layers were isolated within separate virtual machines, thereby ensuring security and isolate sensitive components from public-facing interfaces. This modular design enables maintenance operations, such as incremental updates or system upgrades, to be performed on selected components without inducing downtime for publicly accessible applications. Furthermore, the architecture is intended to facilitate seamless and sustainable scalability by permitting the addition of new servers without requiring the deployment of additional management services on these new hardware assets.

A three-part system architecture, seen in the lower part of Figure 2, is used as the basis for the Sandbox. Three virtual machines (VM) are used on the server, each of which is assigned to a task group. The first VM acts as a management environment, mastering associated services located in other nodes of the cluster. The second VM provides all applications/services that are made available to both internal and external user groups. The third VM operates as the worker node, handles graphical processing (services and drivers), and supports various software agents. Due to the chosen architecture, it is possible to provide GPU-dependent services only on the third VM. However, all VMs can access storage (HDD + SSD), CPU and memory. Experience has shown that these are limited to the respective application area. For example, the worker node requires a particularly large amount of memory for optimal utilization of the graphics card.

Based on the requirements, the following hardware and software components have been identified as essential and are incorporated into the system.

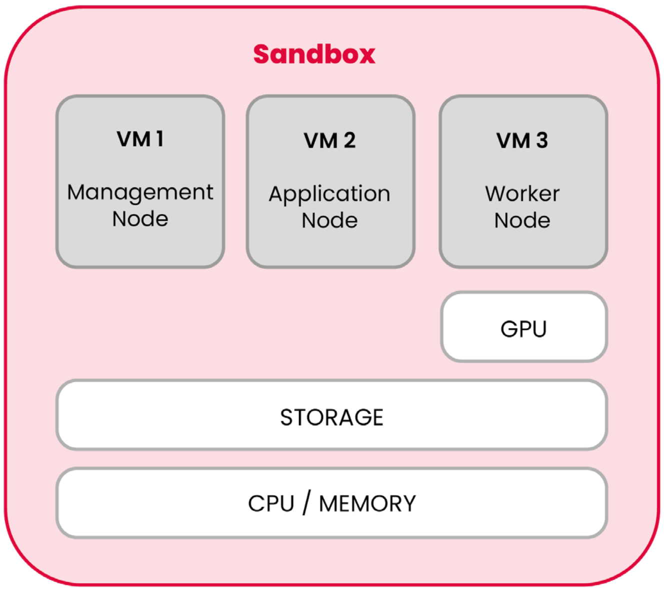

Figure 2: Sandbox Architecture.

Software Components

In the following we identify the core software components and describe the specific software artifacts, which we selected, and why we integrated them into the IKID Sandbox.

This section will not only give a general introduction to each component but also describe the individual software tools and technologies selected. Also, clear and concise reasons behind each choice are provided, highlighting their suitability and alignment with the general objectives and requirements of the project.

Containerization

Component Introduction: Containerization is a software packaging and deployment approach that bundles an application's code, libraries, dependencies, and configuration files into a single, portable unit called a container. These containers run in isolated environments, ensuring consistency across different computing environments and simplifying deployment. Benefits of containerization include enhanced portability, improved resource utilization, faster startup times, streamlined development and deployment cycles, increased scalability and fault isolation. It enables developers to easily move applications between development, testing, and production environments without compatibility issues. Containerization is thus a critical component of deploying software artifacts within the Sandbox.

IKID Sandbox Implementation: In order to provide a comprehensible and thus sustainable development approach, *Kubernetes* (K8s) was chosen, as it offers the benefits of the declarative configuration and access control.

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It provides a robust framework for running distributed systems on a cluster of nodes (servers) at scale. K8s offers a robust platform for deploying and managing containerized applications, offering key advantages such as high availability through automatic Pod replication, effortless scalability via node adjustments, self-healing capabilities that restart failed containers, simplified configuration management with declarative *YAML* or *JSON* files, and the support of a large and active community, providing a wide range of tools and resources.

K8s can be broken down into these key components:

(i) **Pods:** The smallest deployable unit in Kubernetes. A pod represents a single instance of an application, containing one or more containers.

(ii) **Services:** Define how your application is exposed to the outside world. Services provide a stable endpoint for accessing applications running within Pods.

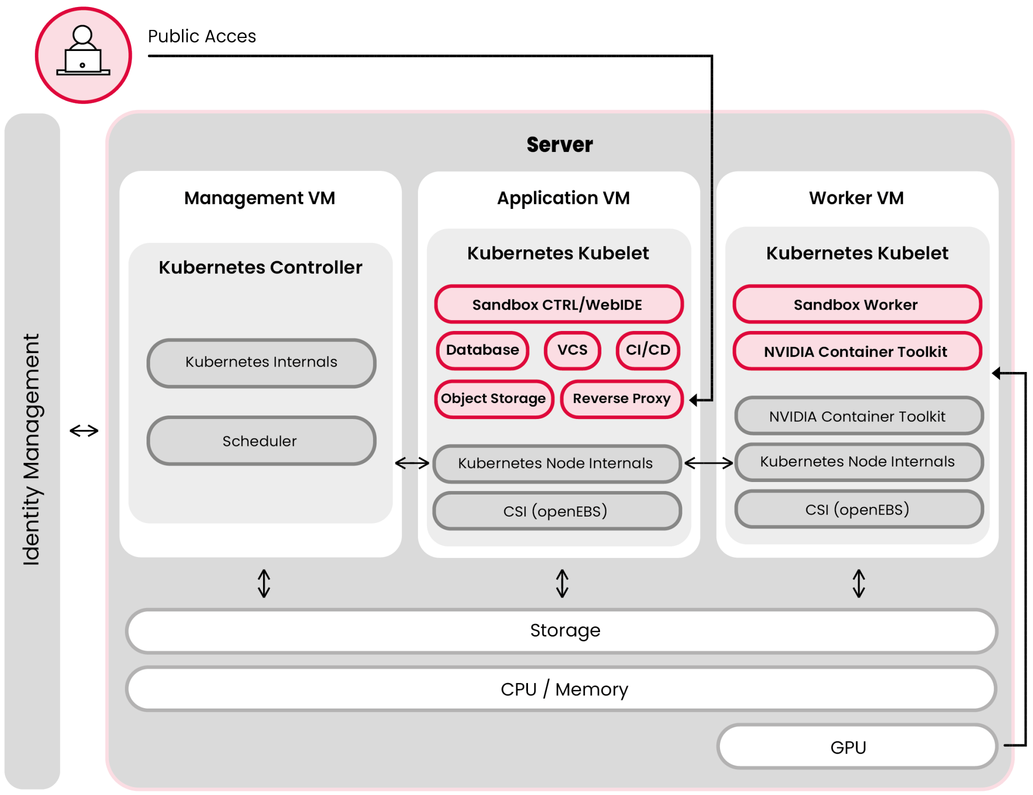

Figure 3: Sandbox Container Architecture.

(iii) **Controllers:** Manage the lifecycle of Pods, ensuring that the desired number of Pods is always running and scaling up or down as needed, such as Deployments, StatefulSets and DaemonSets.

(iv) **Namespaces:** Isolate resources within a cluster, allowing for better organization and access control.

(v) **API Server:** The central control plane for Kubernetes. It receives commands from users and manages the cluster state.

(vi) **etcd:** A distributed, reliable key-value store used by Kubernetes to store configuration and state information.

(vii) **kubelet:** A daemon that runs on each node, responsible for managing Pods and communicating with the API server.

(viii) **kube-proxy:** A network proxy that handles service discovery and load balancing for Pods within the cluster.

Faced with a variety of Kubernetes options, k0s[1] emerged as our chosen solution due to its compelling advantages. k0s distinguishes itself through its lightweight and single-binary nature, simplifying deployment and management. Moreover, k0s boasts a streamlined architecture and minimal dependencies, enhancing security and reducing potential vulnerabilities. These combined benefits make k0s a compelling choice for our Kubernetes implementation, offering both operational efficiency and a robust security posture.

As seen in Figure 3, we decided to deploy a Controller Node on our Management VM, which is connected via OpenID to our Identity Management. The Application VM hosts our foundation software stack which is needed for all further applications and development. Hereby, we include the Sandbox Controller, which automatically starts new isolated WebIDE containers for the users. In addition, we place our database, version control, the CI/CD control plane and object storage, as well as reverse proxy to publish HTTPS endpoints.

Our system architecture leverages three virtual machines: a Management VM hosting the Controller Node, which connects to our Identity Management system via OpenID for secure authentication, and an Application VM containing our foundational software stack. This stack includes the Sandbox Controller, responsible for automatically provisioning isolated WebIDE containers for users, alongside centralized services like the database, version control, CI/CD control plane, object storage, and an ingress/reverse proxy for publishing secure HTTPS endpoints.

The Worker VM is dedicated to deploying on-demand isolated container environments, leveraging the NVIDIA Container Toolkit to enable GPU acceleration for both development tasks and resource-intensive, long-running training processes. This setup provides a robust and efficient environment for streamlined development and deployment processes.

Ingress and Reverse Proxy

Component Introduction: Ingress and reverse proxies are essential components in a containerized environment for managing external traffic to services within a cluster. An Ingress Proxy is a resource that serves as a single entry point for external users to access services within a cluster. It defines rules for routing incoming traffic to these services based on hostnames, paths, and other criteria. Ingress resources allow for secure communication through TLS/SSL termination. Importantly, ingress itself does not handle the actual routing logic but relies on external ingress controllers to implement the defined rules.

A reverse proxy is a server that acts as an agent between clients and backend services, receiving incoming requests and directing them to the appropriate backend server based on configuration rules. Reverse proxies typically handle a variety of functions to optimize application delivery. These include load balancing to distribute traffic across multiple servers, SSL termination to offload encryption processing, rate limiting to protect against overwhelming traffic, and caching to improve performance by storing frequently accessed content. Public-facing endpoints must be equipped with up-to-date, valid certificates to guarantee robust security for the Sandbox when a user transmits data to our services.

IKID Sandbox Implementation: Within the Kubernetes ecosystem, ingress controllers serve as intelligent gatekeepers, meticulously managing external traffic destined for services residing within the cluster. They maintain a constant dialogue with the Kubernetes API server, continuous monitoring for changes to ingress resources – the rules that dictate how external requests are directed. Upon detecting any modifications, the ingress controller dynamically adjusts its configuration, ensuring seamless traffic flow according to the latest directives.

External traffic seeking access to services within the cluster is routed through a Kubernetes service, typically of type „LoadBalancer" or „NodePort," which acts as the entry point to the ingress controller. This setup brings several technical advantages. It provides a centralized point for managing all ingress traffic, simplifying administration and boosting security. Moreover, users benefit from a single, consistent access point, while performance enhancements like load balancing and caching optimize traffic distribution and accelerate response times.

Our infrastructure requires a publicly accessible endpoint on port 443 to facilitate secure communication with deployed services. Utilizing a wildcard domain DNS name (*.sandbox.iuk.hdm-stuttgart.de) grants the flexibility to host multiple services under this subdomain, each accessible via its unique subdomain. The chosen solution for this purpose should not only route traffic to the appropriate service but also handle HTTPS certificates, including automatic renewal, ensuring seamless and secure connections for all services.
While several options like Nginx[2] etc. exist, Traefik emerges

as a strong contender, fulfilling all our requirements. Its proven track record in container environments and comprehensive feature set, including robust routing capabilities and automated HTTPS certificate management, make it a natural fit for our Kubernetes deployment. This prior experience with Traefik[3] gives us the confidence to leverage its capabilities within our Kubernetes infrastructure.

Authentication and Authorization

Component Introduction: Authentication and authorization are fundamental security concepts for controlling access to systems and resources. Authentication verifies the identity of a user or entity, ensuring they are who they claim to be. This is commonly achieved through methods like username/password logins, multi-factor authentication (MFA), or biometric verification. On the other hand, authorization determines the level of access an authenticated user has to specific resources or actions. It defines what a user is permitted to do once their identity is confirmed, like viewing specific data or modifying settings.

In today's web-centric world, OpenID Connect (OIDC) and OAuth 2.0 have emerged as crucial standards for secure authentication and authorization. OAuth 2.0 focuses on delegated authorization, allowing users to grant third-party applications access to their resources on another service without sharing their credentials. OIDC builds upon OAuth 2.0, adding a standardized layer for authentication. It enables users to verify their identity through a trusted identity provider and share their verified information with web applications securely.
By employing these standards, developers can streamline authentication and authorization processes while enhancing security. Users benefit from a seamless experience, as they can often leverage their existing accounts with providers like Google or Facebook to access various web services without creating new credentials for each platform. This reduces password fatigue and improves overall user experience.

In enterprise environments, Lightweight Directory Access Protocol (LDAP) often plays a pertinent role in managing user identities and access permissions. LDAP serves as a centralized directory for storing and retrieving user information, including usernames, passwords, and group memberships. By integrating OIDC or OAuth with an existing LDAP infrastructure, organizations can leverage their centralized identity management system for web-based authentication and authorization. This integration enables single sign-on (SSO) capabilities, allowing users to access multiple applications with a single set of credentials stored in the LDAP directory.

To ensure security and control access to its resources, the Sandbox requires robust authentication mechanisms for both administrative and regular user roles. This multi-tiered authentication process will verify the identity of users, ensuring that only authorized individuals can perform administrative tasks or access sensitive information within the Sandbox.

IKID Sandbox Implementation: Identity management (Identity Provider/IDP), also known as identity and access management (IAM), is a framework of policies and technologies to ensure that the right users have the appropriate access to technology resources. It is one of the core elements in modern enterprise software architecture.

Designing a new software architecture, security aspects should always be considered first. It is the foundation of every other piece of software which will later be deployed.

The current IT infrastructure at HdM offers a limited way to internally authenticate members of the organization by LDAP[4]. For the Sandbox, no specific group or role management is applicable or accessible within the existing infrastructure.

To implement a modern authentication and authorization system with fine-grained control, we require a dedicated Identity Provider (IDP) that fulfills specific criteria. The chosen IDP solution must be enterprise-grade, open-source, well-maintained, and capable of scaling to thousands of users. Crucially, it needs to support LDAP user federation, leveraging our existing infrastructure for both security and a seamless user experience. Essential features include single sign-on, multi-identity brokering (*OpenID Connect, SAML 2.0, Kerberos*), and optional social login capabilities. The IDP should also facilitate user management based on

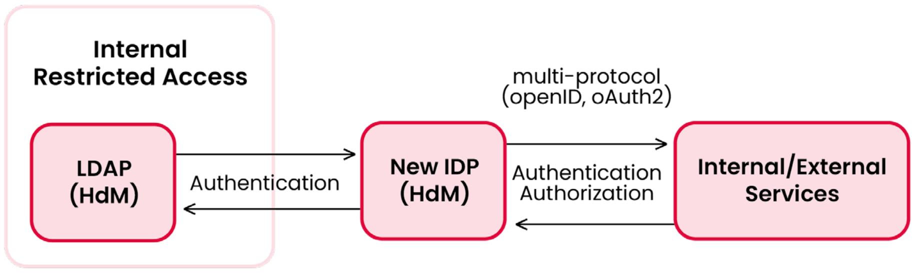
Internal Restricted Access
LDAP (HdM)
Authentication
New IDP (HdM)
multi-protocol (openID, oAuth2)
Authentication Authorization
Internal/External Services

Figure 4: Identity Management Flow.

roles and groups, offer theme customization, and be hosted externally to ensure the stability and availability of our core services. Figure 4 describes the expected authentication/authorization flow for the Sandbox.

After researching several identity providers, including *Gluu*[5], *Authelia*[6], *Keycloak*[7], and *Authentik*[8], we determined that *Keycloak* aligns best with our requirements. *Keycloak* fulfills all our needs, boasts a long history of maintenance and a positive reputation on *GitHub*, and benefits from the support of *Red Hat*[9], indicating its stability and reliability. This combination of comprehensive features, maturity, and strong community support makes *Keycloak* the ideal choice for our identity management needs. Therefore, we deployed customized and connected to the internal LDAP system on a separate external VPS instance at https://auth.iuk.hdm-stuttgart.de.

To enable remote authentication and authorization, we first deployed the k8s plugin kubelogin[10] on our infrastructure. This plugin facilitates a secure approach to remotely managing Kubernetes clusters as seen in Figure 5.

It leverages OpenID Connect (OIDC), a well-established authentication protocol, to establish trust between the user and the Kubernetes API server. This mechanism eliminates the need for pre-shared credentials, mitigating potential security vulnerabilities associated with traditional password-based authentication. By delegating authentication to a trusted OpenID Connect provider, kubelogin enhances the overall security posture of Kubernetes cluster management. Further details and technical specifications regarding the plugin can be found on the GitHub repository. For all services the newly deployed identity management will be used.

Database

Component Introduction: In complex software systems composed of multiple interconnected components like the IKID Sandbox, a centralized database serves as a repository for shared data, ensuring consistency and facilitating efficient data management. By consolidating data into a single location, organizations can eliminate redundancy, improve query performance, and streamline data administration tasks. This centralized approach offers numerous advantages, including enhanced scalability, simplified disaster recovery, and improved data sharing capabilities. Additionally, by providing a single source of truth, it reduces the risk of data inconsistencies and enables more effective data analysis and reporting.

Recognizing that modern applications rely heavily on databases, the Sandbox must include a scalable database solution. This approach will provide a robust and adaptable foundation for applications within the Sandbox, allowing them to store and manage data effectively while accommodating potential growth and evolving requirements.

IKID Sandbox Implementation: While simplistic solutions such as static configuration files and embedded databases like SQLite may suffice for small-scale applications, their limitations become apparent when deployed in distributed environments with substantial user loads such as the IKID Sandbox. To accommodate the demands of our complex systems a centralized, open-source database capable of handling concurrent access and large datasets is indispensable.

Due a comprehensive analysis of the core applications required a database solution that could effectively manage the anticipated data load. *PostgreSQL*[11], owing to

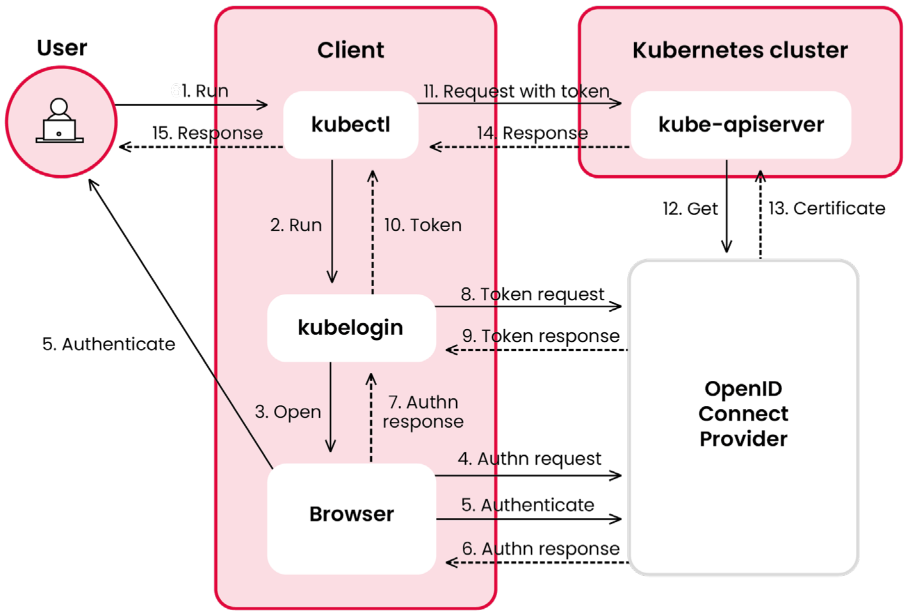

Figure 5: Cluster Authentication.
Source: https://raw.githubusercontent.com/int128/kubelogin/master/docs/credential-plugin-diagram.svg

its robust feature set and widespread adoption, emerged as a suitable candidate. However, deploying and managing PostgreSQL across a cluster environment manually proved to be labor-intensive and error-prone. To streamline this process and enhance reliability, the open-source cluster manager StackGres[12] was selected. This platform offers automated deployment of multiple database servers, coupled with essential functionalities such as backup and restoration, thereby optimizing database management within a complex infrastructure.

For all services which require a database, our *StrackGres* deployment is utilized.

Version Control System

Component Introduction: Version control (also known as revision control, source control, or source code management) is a class of systems responsible for managing changes to computer programs, documents, large websites, or other collections of information. A version control system (VCS) is a software tool that tracks and manages changes to code over time, acting as a time machine for your projects. By recording every modification, VCS enables developers to revert to previous versions, collaborate efficiently, and gain valuable insights into code history. Git[13], the industry standard, offers a distributed approach, allowing teams to work independently and merge changes seamlessly. For independent individual activities, developers could use branching features to stage commits and have a chronological commit history. Beyond code, VCS can also manage other project files but is most common for code tracking.

A robust version control system is essential for the Sandbox to support internal and external software development efforts effectively. This system will serve as a central repository for code, and other project artifacts, facilitating collaboration, tracking changes, and ensuring code integrity throughout the development lifecycle.

IKID Sandbox Implementation: To establish a code hosting and management platform accessible to both our internal Sandbox team and external students or lecturers, we require an open-source solution that meets specific criteria. The ideal platform should be well-regarded within the open-source community, support OAuth2 authentication for secure access, and offer a modern and intuitive web interface. While optional, integrated package management capabilities, including support for container and Python repositories, would further enhance the platform's utility.

Git meets these requirements. It is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers who are collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed workflows. While a headless Git server provides version control, it lacks integrated user management. GitLab[14], though feature-rich, presents licensing uncertainties for its advanced features. Therefore, we opted for Gitea[15], a rapidly growing open-source platform with a clear licensing model. Its adoption by reputable non-profit projects like Codeberg[16] further promotes Gitea's capabilities and suitability for our needs, providing a robust and transparent solution for our code hosting requirements.

A significant advantage is its integrated universal package registry, allowing us to host container images and Python packages seamlessly without requiring additional infrastructure or management overhead.

Package Registry

Component Introduction: A package registry is a centralized repository for storing and managing reusable software packages. Developers can publish, discover, and install packages using these registries, accelerating development by leveraging pre-built components. They function as a marketplace, allowing developers to share code and collaborate effectively. While package registries offer benefits like code reuse, dependency management, and faster development cycles. Beyond traditional software packages, modern package registries also support other artifact types, including Python libraries and container images. This expanded scope enables developers to manage a broader range of project dependencies, streamline deployment processes, and foster a more comprehensive development ecosystem.

IKID Sandbox Implementation: The IKID Sandbox requires a dedicated package registry to effectively manage and deploy its other software components. This registry will serve as a centralized repository for storing and distributing customized containers, self-made packages, and other software artifacts specific to the Sandbox ecosystem. As mentioned in the previous paragraph, the package registry was deployed using Gitea.

Continuous Integration and Continuous Delivery/Deployment

Component Introduction: CI/CD, short for Continuous Integration and Continuous Delivery/Deployment, revolutionizes software development by automating the entire software lifecycle. Continuous Integration (CI) ensures a robust codebase by frequently merging code changes, triggering automated builds and tests to catch integration errors early on. Continuous Delivery (CD) picks up from there, automatically preparing software releases for deployment, whether to staging environments for further testing or directly to production. This guarantees that software is always in a deployable state, ready to rapidly deliver new features.

The beauty of CI/CD extends beyond traditional software to complex machine learning workflows. Consider the lengthy process of AI model training. CI/CD pipelines can automate data preparation, model selection, training execution, and even performance evaluation. This automation carries through to deployment, integrating the trained model into production applications, and establishing ongoing monitoring and retraining as needed. This not only accelerates the deployment of AI models but cultivates a continuous improvement cycle, where rapid feedback on code, data, and model performance leads to faster development, quicker issue resolution, and ultimately, a more robust and efficient machine learning process.

To streamline and automate its software development and machine learning workflows, the Sandbox environment needs to implement robust CI/CD.

IKID Sandbox Implementation: To increase the speed and the quality of software development, a universal CI/CD pipeline needs to be provided which offers an easy syntax for build recipes and is highly extensible such that a GPU can be included in order to run long running AI trainings in an isolated environment. The integration (automatically deployments within the cluster) is not needed for our particular use case.

We have evaluated continuous integration/continuous delivery (CI/CD) platforms like Travis CI[17], Jenkins, and Drone[18]. The appeal of container-based build pipelines led us to strongly consider Drone. However, due to a licensing change in Drone, we opted for its community-driven fork, Woodpecker[19], as our CI/CD solution.

Our adoption of container-based build steps, coupled with GPU support, allows us to leverage existing containers from Dockerhub[20] or GPU-enabled solutions from providers like NVIDIA[21] [22]. These solutions readily offer frameworks like TensorFlow[23] and PyTorch[24], streamlining our training process.

Graphics Processing Unit

Component Introduction: GPUs excel in machine learning due to their parallel processing power. Thousands of cores allow simultaneous operations on large datasets, dramatically accelerating training and inference. Their optimization for matrix operations, the cornerstone of machine learning algorithms, delivers significant performance gains. Furthermore, high memory bandwidth ensures swift data transfer, crucial for handling massive datasets.

The NVIDIA A100 takes these capabilities a step further with its advanced time-slicing features. This allows the GPU to partition its resources and execute multiple tasks concurrently, maximizing utilization and ensuring efficient processing even for diverse workloads.

Moreover, the A100's vGPU (virtual GPU) capability enables the division of a single physical GPU into multiple isolated virtual GPUs. Each vGPU operates with dedicated resources, providing performance predictability and isolation for multi-tenant environments, further amplifying the A100's utility in demanding machine learning scenarios.

By leveraging the time-slicing and vGPU capabilities, a single NVIDIA A100 can effectively serve multiple users with concurrent machine learning capabilities. This simultaneous access allows for efficient resource allocation and provides each user with a dedicated and isolated environment for their specific machine learning tasks.

IKID Sandbox Implementation: The graphics card driver is essential for high-performance machine learning, unlocking a range of functionalities. It acts as a critical bridge

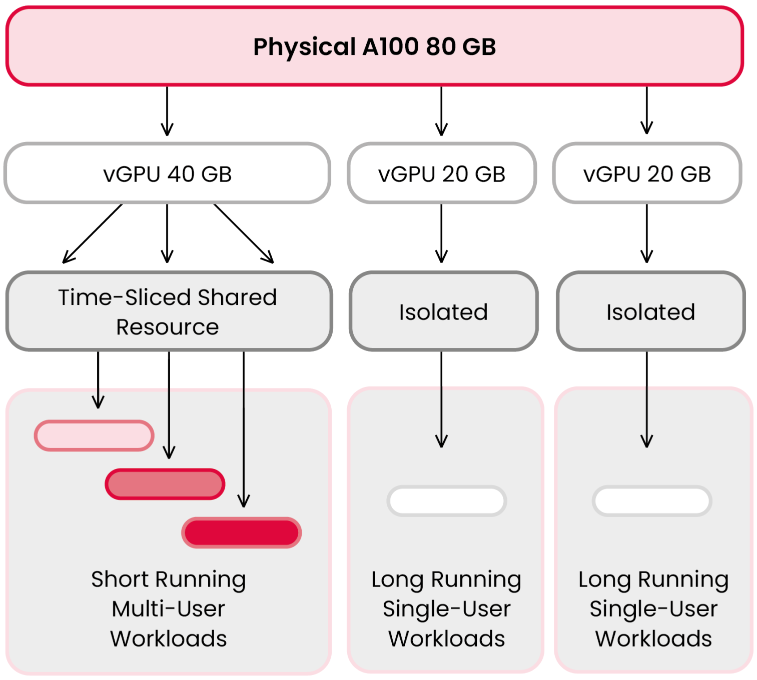
Physical A100 80 GB
vGPU 40 GB
vGPU 20 GB
vGPU 20 GB
Time-Sliced Shared Resource
Isolated
Isolated
Short Running Multi-User Workloads
Long Running Single-User Workloads
Long Running Single-User Workloads

Figure 6: Virtual GPU Architecture.

between software and hardware in our Sandbox, enabling complex computations, optimized memory management, and accelerated data processing, ultimately driving efficiency and speed in machine learning tasks.

We created a sustainable GPU configuration that serves both basic and advanced users. To achieve this, we divided our A100 GPUs into two distinct resource pools using NVIDIA Container Toolkit (cf. Figure 6). Advanced users, primarily those running continuous integration (CI) and long training pipelines, receive dedicated access to fixed-size virtual GPUs (vGPUs) of 2x20GB. This guarantees them consistent, predictable performance for their resource-intensive tasks.

Basic users, typically working on the playground or executing short-running GPU tasks within WebIDE's, benefit from a more dynamic allocation. A single 40GB vGPU is time-sliced into ten smaller portions, allowing multiple users to share the resource efficiently. This approach optimizes GPU utilization by catering to the varying demands of different user groups.

Web-based Integrated Development Environment

Component Introduction: A Web-based Integrated Development Environment (WebIDE), is a powerful, open-source development environment designed with ease-of-use in mind. Hosted on a web browser, it provides a clean and intuitive interface for building and running small applications or demonstrating specific use cases. Its true strength lies in its extensibility and customization options. Developers can leverage a rich ecosystem of existing plugins or readily create their own to tailor the WebIDE to their exact requirements. This combination of user-friendliness and extensibility makes a WebIDE an ideal platform for a wide range of users. Beginners can quickly grasp the basics and start developing, while experienced users have the flexibility to integrate advanced functionalities and create highly specialized development environments. Whether used for rapid prototyping, educational purposes, or showcasing specific technologies, a WebIDE's adaptability ensures a powerful and customized user experience.

To further enhance its capabilities and broaden its appeal, a WebIDE should provide both novice and experienced users with straightforward, remote access to machine learning tools. This accessibility will empower a wider audience to explore, experiment with, and leverage the power of machine learning, fostering innovation and spreading access to this transformative technology.

IKID Sandbox Implementation: After careful evaluation of web-based development environments, including *Visual Studio Code*[25], *AWS Cloud9*[26] and *JupyterHub*[27]. Finding a WebIDE to accommodate users with a wide range of skill levels presents significant challenges. Out of all competitors, *JupyterHub* was selected as the foundation for this project. It has a user interface that prioritizes simplicity, clarity, and extensive customization options. Another important advantage is a fully open-source platform that offers a clean and intuitive interface for developing, running small applications, and writing formatted text.

A WebIDE serves as a controlled execution environment for use case demonstrations that are available to users with different skills. The different skill groups are defined on the project requirements. Pre-defined resource sharing enables efficient experimentation by minimizing configuration overhead. Utilizing *JupyterHub's* extensible architecture, the WebIDE implements a plugin system, enabling the utilization of pre-existing extensions and the development of functionalities tailored to specific requirements.

Object-Storage

Component Introduction: Object storage is a specialized data storage architecture designed to handle the ever-growing volume of data, such as multimedia files, backups, and large datasets. Unlike traditional file systems that rely on hierarchical structures, object storage organizes data as discrete units called objects, each identified by a unique key. This approach eliminates file path limitations and enables massive scalability and cost-effectiveness, especially for storing infrequently accessed data.

The key advantages of object storage lie in its inherent scalability, cost efficiency, durability, and accessibility. Its ability to horizontally scale by simply adding more storage resources makes it ideal for handling exponential data growth. Furthermore, the use of data replication ensures high availability and resilience against hardware failures. Finally, accessibility through RESTful APIs enables seamless integration with modern applications and ensures data is readily available wherever and whenever needed.

To align with modern application development practices and to leverage the benefits of scalable storage solutions, the Sandbox environment needs to incorporate support for object storage.

IKID Sandbox Implementation: To host backups, large structured and unstructured data, a S3 API[28] compatible object storage is needed. The advantage of having S3 API compatibility is that it allows for seamless integration with numerous applications that already support this widely-used interface. Furthermore, implementing a system with multiple distributed drives enhances data protection by increasing fault tolerance. This means that even if one drive fails, the data remains accessible and protected from loss.

MinIO[29], a leading and cutting-edge object storage solution, is purpose-built for containerized environments like Kubernetes and distributed deployments. Its robust feature set includes OpenID support, enabling advanced user management capabilities within its web-based interface.

To ensure high availability and prevent data loss despite having only one server, we've opted for a single-node, multi-drive MinIO deployment. This configuration utilizes four drives, with two hosted locally on the server and two mounted from an external network attached storage (NAS) via the network file system protocol (NFS). While MinIO's distributed mode typically requires four instances with a tolerance for two failures, our setup leverages multiple drives to achieve similar redundancy. This ensures continued read-only access to data, even in the event of two drive failures.

The storage solution will serve multiple roles, encompassing daily backups of database dumps and Gitea related data (excluding Git Large File Storage (LFS) and packages). Additionally, the object storage provides versatile file sharing capabilities through both command-line and web-based interfaces.

Additional Tools

Two supplementary services have been integrated to expand the data pool. The first is a CLI-based REST API facilitating binary sharing, while the second is a graphical user interface to make file sharing among diverse user groups available.

A straightforward ShareCLI REST API (rest2s3[30]) was implemented to facilitate the upload of binary data to an object storage. Uploaded files are automatically deleted after a certain period based on the predefined expiration policy.

ShareUI, is a streamlined file-sharing server that offers temporary file storage with expiration based on download count or time. Functionally similar to the discontinued Firefox Send, it distinguishes itself by restricting uploads to administrators. This architecture empowers individuals and organizations to effortlessly share files while optimizing disk space and maintaining granular control over file distribution. An API facilitates programmatic interaction, and cloud storage integration with AWS S3 is supported as an alternative to local storage. Customizable interfaces can be created using HTML and CSS, and encryption, including end-to-end options, is available. ■

Notes & Links

1 https://k0sproject.io
2 https://nginx.org
3 https://www.traefik.io
4 https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol
5 https://gluu.org
6 https://www.authelia.com
7 https://www.keycloak.org
8 https://goauthentik.io
9 https://www.redhat.com
10 https://github.com/int128/kubelogin
11 https://www.postgresql.org
12 https://stackgres.io
13 https://git-scm.com
14 https://www.gitlab.com
15 https://www.gitlab.com
16 https://codeberg.org
17 https://www.travis-ci.com
18 https://www.drone.io
19 https://woodpecker-ci.org
20 https://hub.docker.com
21 https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch
22 https://catalog.ngc.nvidia.com/orgs/nvidia/containers/tensorflow
23 https://www.tensorflow.org
24 https://pytorch.org
25 https://code.visualstudio.com
26 https://aws.amazon.com/de/cloud9
27 https://jupyter.org/hub
28 https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html
29 https://min.io
30 https://github.com/CSpecht/rest2s3

04
Conclusion and Outlook

This white paper has outlined the overall technical design of the IKID Sandbox, a platform crafted to support the responsible exploration and application of AI within higher education. By integrating a carefully curated selection of open-source technologies, the Sandbox provides a secure, scalable, and user-friendly environment for both novice and experienced users alike. The modular architecture, centered around Kubernetes for containerization, ensures flexibility and adaptability, allowing the system to readily accommodate future growth and evolving needs.

Key design choices, such as the implementation of robust authentication and authorization mechanisms through identity management, coupled with secure communication protocols and isolated containerized environments, emphasize security aspects within the Sandbox. The strategic utilization of a centralized database, efficient object storage, and a fully flavoured version control system further contributes to a robust and reliable platform.

Moreover, the Sandbox enables user experience without compromising functionality. The intuitive WebIDE caters to diverse skill levels and tools like ShareUI/ShareCLI allow file sharing and collaboration. By prioritizing accessibility and ease of use, the Sandbox empowers a wider audience to engage with AI technologies, fostering innovation and spreading access to this transformative field.

The IKID Sandbox, with its well-defined architecture, carefully selected technologies, and commitment to security and user experience, is poised to become an invaluable resource for responsibly teaching and learning about AI. By providing a controlled yet versatile environment for exploration, experimentation, and collaboration, the Sandbox lays a strong foundation for fostering a deeper understanding of AI and its potential impact on society.

The IKID Sandbox has been designed not as a static entity but as a dynamic platform with a sustainable and extensible architecture, primed for ongoing evolution and adaptation. This inherent flexibility ensures its continued relevance and utility beyond the initial project phase. As the AI landscape continues to evolve at an electrifying pace, so too will the Sandbox, readily incorporating new tools and technologies to remain at the forefront of AI education. During development, we have remained agile and responsive to the rapid advancements in the field of AI. We extended our initial work packages to prioritize the integration of technologies like modern state-of-the-art AI tools, rather than solely focusing on the initial scope. This proactive approach is reflected in the integration of cutting-edge technologies, including Large Language Models (LLMs) for natural language processing tasks, facial recognition systems for computer vision applications, audio transcription tools for speech analysis, and reinforcement learning environments for exploring agent-based learning paradigms.

This commitment to continuous improvement ensures the IKID Sandbox remains a valuable resource for the foreseeable future, empowering educators and learners to confidently navigate the ever-expanding frontiers of AI. By providing a flexible and future-proof platform, the IKID project paves the way for a new generation of AI practitioners equipped with the knowledge and tools to shape the future of this transformative technology responsibly. ■

Building the IKID Sandbox

Imprint

The whitepaper series on the interdisciplinary research project "IKID: Interdisciplinary AI Teaching" is dedicated to the emerging field of artificial intelligence in university didactics. A total of six issues deal with AI-relevant competences, interdisciplinary AI teaching concepts, exemplary teaching methods and didactic forms of teaching. Furthermore a technical infrastructure is introduced in which students can learn the relevant skills and abilities.

Editor:
Marcel Schlegel

Design and Layout:
Michelle Stegner and Marcel Schlegel

Contact:
Prof. Dr. David Klotz – klotzd@hdm-stuttgart.de
Hochschule der Medien Stuttgart, Nobelstraße 10, 70569 Stuttgart

Publisher:
Digipolis Verlag
Sina Klauke, Tramweg 8, 77966 Kappel-Grafenhausen
kontakt@digipolis-verlag.de
www.digipolis-verlag.de

https://doi.org/10.70481/rck1-6tm3

HOCHSCHULE
DER MEDIEN