Research online Publishing
www.ronpub.com

Open Access

Open Journal of Cloud Computing (OJCC)
Volume 8, Issue 1, 2023
http://www.ronpub.com/ojcc
ISSN 2199-1987

Closing the Gap between Web Applications and Desktop Applications by Designing a Novel Desktop-as-a-Service (DaaS) with Seamless Support for Desktop Applications

Christian Baun, Johannes Bouché

Faculty of Computer Science and Engineering, Frankfurt University of Applied Sciences,
Nibelungenplatz 1, 60318 Frankfurt am Main, Germany
christianbaun@fb2.fra-uas.de, johannes.bouche@fb2.fra-uas.de

Abstract

An increasing transformation from locally deployed applications to remote web applications has occurred for about two decades. Nevertheless, abandoning established and essential Windows or Linux desktop applications is in many scenarios impossible. This paper describes and evaluates existing Desktop-as-a-Service solutions and the components required for developing a novel DaaS. Based on the conclusions and findings of this analysis, the paper describes a novel approach for a Desktop-as-a-Service solution that enables, as a unique characteristic, the deployment of non-modified Linux and Windows applications. The interaction with these applications is done entirely through a browser which is unusual for remote interaction with Windows or Linux desktop applications but brings many benefits from the user's point of view because installing any additional client software or local virtualization solution becomes unnecessary. A solution, as described in this paper, has many advantages and offers excellent potential for use in academia, research, industry, and administration.

Type of Paper and Keywords

Regular research paper: *Desktop-as-a-Service, DaaS, Cloud Computing, Virtualization, Container*

1 Introduction

This paper introduces an architecture blueprint for a novel Desktop-as-a-Service (DaaS) solution that can be deployed in public and private cloud scenarios, which is an essential criterion for many possible application scenarios. Furthermore, the architecture enables the deployment and usage of unmodified Linux and Windows applications in the same way as web applications. All interaction with the applications imported into the DaaS is done via a users web browser. No other additional client software besides that browser is required in this architecture. Unlike similar efforts, this solution is intended to integrate both Windows and Linux applications.

This document is structured as follows. First, section 2 explains why DaaS is a relevant component in current and future IT landscapes and derives requirements for a state-of-the-art DaaS approach. Section 3 discusses related work on DaaS solutions from an academic perspective, whereas section 4 describes and analyzes existing DaaS projects. Section 5 presents an identification and analysis of possible components required for developing a novel DaaS. These components include server-side services for remote access to the user

interfaces of native applications for Linux and Windows, proxy technologies that convert the graphical output into a browser-accessible representation as well as a collection of tools for orchestrating, monitoring, and managing containers, virtual machines, or the underlying hardware. The results of this analysis are combined into a proposal for a state-of-the-art DaaS architecture as presented in section 6. Finally, section 7 discusses conclusions and directions for future work.

2 Motivation for Using DaaS and Requirements for a New DaaS

A modern DaaS has the potential to bridge traditional local software and traditional Software as a Service (SaaS) solutions, providing useful capabilities in particular for mobile or shared workplaces or at the home office. The following subsections discuss those aspects in more detail and derive essential requirements for a suitable DaaS solution.

2.1 DaaS as a Bridge Between Local Desktop Software and SaaS

Locally running desktop software has been a staple of personal computing, particularly with office suites such as LibreOffice (OpenOffice) or Microsoft Office. By running locally, such software is well suited to rich user interfaces that can quickly react to user input. Data is stored locally, which can have privacy and security advantages. Once installed, software can be used indefinitely, leaving aside the issue of license checks.

All of this presents a degree of digital autonomy for users. Nevertheless, there are also disadvantages to this model, for example, that software is specific to an operating system or even the underlying hardware and that management tasks such as backups, data synchronization, or security updates must be handled separately. For these reasons maintaining software deployments across a large fleet of devices can be challenging and error-prone.

Software as a Service (SaaS) provides an alternative that provides web-based software via some vendors' servers to provide end users access via their web browser.

While web browsers initially displayed static documents, they have become powerful application platforms in their own right. In the 2000s, they gained the ability to perform complex network operations in the background while still making it possible to handle large amounts of user input without noticeable network latency. A typical application of this early web application era is the original Gmail (2004). During the 2010s, browsers gained additional capabilities that approach those of desktop applications, such as Web Real-Time Communication (WebRTC), Web Graphics Library (WebGL), and advanced web applications. Combined with software frameworks like Angular or React, it is often easier to create a web application than a native desktop application.

SaaS has different advantages for service providers and end users compared to traditional desktop software. Service providers can evolve SaaS more quickly since changes become immediately available to all users simultaneously and without users having to upgrade from old versions explicitly. Since the service's value is provided via web servers, software piracy is less often a problem, and subscription models can be enforced. End users enjoy the advantage of less reliance on their local computers. Any device with a web browser can access SaaS, and since data resides "in the cloud", it is effectively synchronized or backed up immediately. For organizations with centralized identity management, rolling out access to a SaaS is more effortless than deploying locally installed software on devices.

However, there are also drawbacks and limitations to this SaaS approach. While a SaaS vendor is potentially more capable of securing that service, end users must trust that vendor completely. Since the data resides in the SaaS vendor's servers, there is also a risk of vendor lock-in. More immediately, end users will notice that such applications generally require a network connection, which can limit when and how such SaaS applications can be used. Finally, SaaS can only be used if such a service exists. While some types of software now have cloud-based pendants, this is not universally the case, and it is impossible to lift and shift all desktop workloads to SaaS providers.

A variety of hybrid approaches bridge the gap between desktop software on the one hand and SaaS/web applications on the other.

Much work has been done to bring the web into the desktop. For example, whole operating systems like ChromeOS and Firefox OS are built around web technologies. Due to frameworks like React Native or Electron, it is also fairly common to develop native applications (mobile or desktop) using web technologies.

Compared to this, bringing the whole desktop to the web is still a rare use case. Some desktop applications have been ported to the web platform, for example, by using Emscripten [55]. An example of this is Collabora Online [14], which is a web-based port of LibreOffice. There have been some attempts to bring the "desktop metaphor" (user interface with multiple floating windows) to the web, but they have been greeted with limited success. Examples of such web desktops are collected in section 4.1.

Proper Desktop as a Service (DaaS) solutions also exist, which run a desktop environment on a server and

then provide remote access via protocols like Virtual Network Computing (VNC) [38] or Remote Desktop Protocol (RDP). This paradigm can provide SaaS-style access to traditional desktop applications, decoupling the end user's physical device from the software they can run.

2.2 DaaS in the Workplace

Due to the combination of advantages from local desktop software and SaaS, the ability to run on organization-controlled hardware can provide several benefits for the service provider and users. On the one hand, virtualization provides abilities for compartmentalization and individualization of specific environments, leading to more usable services. On the other hand, the encapsulation of applications reduces the integration and management efforts leading to cheaper and more efficient services.

In today's society, and especially in the light of the global Covid 19 pandemic, there is an increased need for workforce mobility. Consequences include high demand for flexible, location-independent workspaces that can be used with any device and still offer high data security and privacy. In addition, because of the trend that has existed for several years among schoolchildren, students, scientists, and employees in companies (especially in creative professions) to be able to work with their own devices (BYOD - bring-your-own-device), there is an increasing need to support multiple mobile devices with heterogeneous hardware and software equipment that is not only owned by the employees but also managed by them [1, 9, 43].

As the BYOD trend grows, so does the need for desktop virtualization solutions that provide unified desktops with identical functionality and mask the heterogeneity of the operating systems and underlying hardware. However, providing a powerful and user-friendly desktop virtualization solution is a financial and, due to the high complexity of the available solutions, also a personnel challenge for many schools and universities, which in the vast majority of cases are under continuous pressure to save money, requiring cost-effective solutions. In traditional environments, organizations allowing BYOD policies are typically required to additionally protect their infrastructure from unintended interaction with non-organizational devices.

The DaaS approach can, by definition, minimize such efforts in most cases while still offering users to utilize their own devices in a trustworthy environment. This is achieved by leveraging the intrinsic approach to utilize immutable deployments within the organizations infrastructure and as a base for the virtualized user application. By doing that, a well-defined base for further processing is provided to the user. This eliminates a substantial amount of organizational effort to troubleshoot user-specific software installations and configurations on the one hand. On the other hand, instead of protecting against the unpredictable behavior of heterogeneous devices, all user devices, only individual and well-defined services, must be protected. Therefore, DaaS solutions have substantial advantages compared to other traditional approaches, especially in shared workplaces where BYOD policies are well accepted.

Furthermore, it could bridge the gap in several mobile scenarios where users often switch between multiple workplaces or computing devices and could alternatively stick to their own custom devices. This might conveniently allow platform-independent usage of any recent and legacy applications for different operating systems hosted by the DaaS. Combined with the usage of 'Thin Clients' or other hardware standardization, this might reduce heterogeneity as well as the total amount of managed devices dramatically, again allowing further cost reductions.

2.3 Requirements for a New DaaS Based on Insights from the Status Quo.

Requirements for the system can be viewed from an administrator perspective or an end-user perspective.

For end users, access to the DaaS applications must be simple and seamless, without requiring the installation of additional client software. In particular, an up-to-date web browser should be sufficient. Applications should be integrated as far as possible into the users' systems, for example, by forwarding audio and video streams, by showing native notifications, and by making it easy to up- and download files.

Applications should also be able to interact with each other, for example, by exchanging files and opening documents in another DaaS-provided application. Using applications alongside each other should be possible, even when they run on different platforms (e.g., Windows and Linux). Users should not have to click through a desktop-style interface to launch applications. However, they should be able to treat them as web pages – with one desktop application per browser tab and the ability to bookmark applications.

From a administrator's perspective, such solutions should be simple to set up and to maintain. The solution should be cloud-ready but still enable installation on local servers to realize the potential security benefits. Packaging existing Windows and Linux applications without modifying them should also be possible. The underlying components should be stable and ideally be based on actively maintained Open Source software.

2.4 Characteristics of DaaS in Comparison with Other Cloud Service Categories and VDI

The National Institute of Standards and Technology (NIST) defines the three main service models in which cloud-based environments are typically provided as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [31].

By that definition, SaaS solutions usually enable users to utilize applications maintained by the provider and his infrastructure. In contrast, PaaS solutions offer the opportunity to deploy custom applications to that infrastructure. An IaaS solution additionally allows users to provision major components of the operating systems, such as networks, storage, or other fundamental computing resources.

According to these basic definitions, a Desktop as a Service (DaaS) solution can be described as a specific kind of PaaS approach in which users are provided with either a tailored or generic Virtual Desktop Infrastructure (VDI) [17] as well as the ability to host custom user applications by using a platform-independent client such as a web browser. Besides, all other characteristics in the NIST definition should also be applicable. A DaaS service is, therefore, an on-demand self-service provided through broad network access, might offer resource pooling, rapid elasticity, and a measuring service, and can be deployed as a public, private, hybrid, or community cloud.

In that sense, the DaaS approach is mainly driven towards the inherent VDI integration and the hosted application's integration. It promotes optimal usage of components and technologies along the whole axis of provisioning. Depending on the hosted application, this usually includes the integration of capable hardware devices, the maintenance of several virtualized or containerized operating systems and their related VDI's, the utilization of efficient network protocols and distributed storage, as well as mechanisms to control such instances in a standardized way in order to install, configure and run one or multiple applications.

In addition, other paradigms and principles might be adopted, which are more focused on decentralizing individual computing resources such as specific Offline-First strategies [2, 29] or Dew-Computing [37, 44, 51], Edge-Computing [40, 42, 49] and Fog-Computing [12, 21, 54]. Although they generally improve the overall user experience with better performance and reduced latencies, integrating them for all possible application scenarios is not always feasible. Nevertheless, integrating such related paradigms is highly beneficial but not necessarily a characteristic DaaS criterion.

3 Related Work

DaaS has been a well-known service category since the emergence of cloud computing. Still, it got much less attention in research and literature than infrastructure (IaaS) and platform services (PaaS).

Celesti et al. [10] implemented in 2016 a DaaS using OpenStack and analyzed the characteristics and performance aspects of using noVNC, SPICE, and Apache Guacamole as solutions for providing access to the desktop via a browser. One focus of the paper is the redirection of the sound interface of the virtual desktop. The paper's authors conclude that Guacamole in combination with the protocol RDP is the best solution.

Magana et al. [30] compared in 2019 the most popular remote desktop protocols and software implementations. The authors analyzed PCoIP used in the Amazon WorkSpaces, Microsoft RDP, TeamViewer, VNC, and Citrix Independent Computing Architecture (ICA). In the paper, the network transfer rate and its relation to the quality experienced by the DaaS user are evaluated by assuming three scenarios: using an office software suite, web browsing, and video streaming.

Nakhai and Anuar [33] evaluated in 2017 the resource consumption of virtual desktops on top of several variants within the Windows operating system family. The paper focuses on memory usage, CPU, and application response time while different workloads are simulated using popular applications like Outlook, Word, and Adobe Acrobat Reader, as well as music and video streaming.

Deboosere et al. [16] propagated in 2012 DaaS as a role model for using energy-efficient and mobile thin-client devices. The authors propose the need for a novel, objective metric that focuses on the user experience. Furthermore, the authors propose extending the existing cloud management algorithms so virtual desktops can be relocated from overloaded data centers to ones with a lesser load.

Dernbrecher et al. [17] published 2013 a taxonomy for desktop virtualization in which common characteristics are specified. The publication compares and categorizes traditional and virtualized setups with each other, refers to relevant literature, and defines fundamental technical terms concerning VDI.

In contrast to the related works in this section, we propagate an architecture of free software components for implementing a DaaS that supports all sorts of applications developed for Linux and Windows operating systems, which sets focus on each individual application rather than the whole desktop environment.

4 Analysis of Existing DaaS Projects

In order to give an overview over existing DaaS projects in the problem area, all mentioned projects are categorized into:

- so-called web desktops, based on web languages or web technologies such as HTML, JavaScript, Silverlight, and PHP, and
- DaaS based on container virtualization.

Listing and describing all existing DaaS projects based on web languages is not very helpful. Instead, this analysis primarily aims on projects that may be interesting for developing a novel DaaS by concentrating on essential features, relevance, up-to-dateness, and usability.

4.1 Web Desktops

This section summarizes existing ”web desktop” or ”webtop” projects, i.e., software that tries to mimic a desktop experience in the web browser or tries to provide a web-based operating system. However, none provides a feasible path towards accessing desktop applications through a browser. Most of these projects are inactive and outdated, and the remaining candidates focus purely on implementing desktop-like functionality rather than on integrating existing applications.

In the context of implementing a modern DaaS solution, none of these projects provide technology that could be reused. At most, they illustrate that it is not necessary to use native desktop applications for everything. For example, the proposed DaaS solution might use web applications to provide functionality like a file explorer or text editor to the user.

The eyeOS project (2005) [28] simulated a desktop environment using HTML, PHP, JavaScript, and MySQL. It found some use in schools. The software was open source under an AGPLv3 license until version 2.5 in 2011. Telefonica acquired the project in 2014 and expanded it to support running Linux and Windows applications in the browser. With Open365, an AGPLv3-licensed variant was published, which supported desktop software such as LibreOffice, Mozilla, GIMP, and some KDE applications. However, development stopped in 2016. Oneye was an attempt to revive the Open Source eyeOS project, but it was discontinued in 2017. Using components from these projects is impossible as they are severely outdated.

SilveOS mimics the look and feel of Windows 10. It integrates other SaaS web applications such as Google Docs, but cannot integrate desktop applications and does not provide any user management. The source code is available in a public github repository [20].

OS.js provides a web interface similar to the Windows or MacOS user interface. It is Open Source under the BSD-2-Clause license, and has an active developer community. It offers some interesting features such as persistent configuration, authentication, and file system, but cannot integrate native desktop applications.

CorneliOS implemented a web-based operating system with its own application toolkit, a virtual file system, user management, and a content management system. However, the last release of the GPL-licensed software was in 2015, and the source code is no longer available on its SourceForge page. In any case, it would not have been possible to integrate native desktop applications into this system.

The daedalOS software implements a DaaS in the browser that imitates the look of a modern Windows operating system desktop. The software is written in JavaScript and is free software according to the MIT license. The project is under active development and has a very appealing visual appearance. Unfortunately, it is one of the numerous existing solutions that run exclusively in the client's browser and only simulate a desktop. To extend the range of functionality, the developer uses numerous existing JavaScript solutions such as BoxedWine, EmulatorJS, js-dos, Monaco Editor, jspaint, pdf.js, xterm.js, and Webamp. The project does not have a dedicated server component with user administration or data management, and does not provide the ability to host native Linux or Windows applications.

The DaaS solution Online Operating System (Online OS) is free software written in JavaScript and DHTML. Communication via client and server is implemented using Ajax. The appearance of this solution looks like a Windows desktop. The last published version (v1.3.01) is from 2008. The source code can no longer be found. The platform enables the development and execution of own applications online, directly in the DaaS solution, providing a so-called Collaborative Document Management Module (OOS.CDM) as well as a virtual file system.

The UniverseOS project differs from other web desktops in a way that its focus was on enabling anonymous collaboration on encrypted files for journalists and for example human rights activists [36, 48]. It was developed from 2013 until 2015 and is since inactive, but the source code is still available in a public repository. Due to its specialized nature, it does not lend itself to integrating native desktop applications.

4.2 DaaS Projects Based on Container Virtualization or Similar Technologies

Container virtualization and hypervisor-based virtualization, e.g., via the Kernel-based Virtual Machine (KVM) [6, 25] and the Quick Emulator (QEMU) [5], can be useful for running desktop applications in an encapsulated manner. Enriching the containers or virtual machines with a protocol for remote access, such as VNC or RDP, already provides the essential basic technologies for a modern DaaS. Some projects like linuxserver/webtop, Ulteo OVD, and Kasm Workspaces implement comparable solutions on a smaller scale.

4.2.1 linuxserver/webtop

The linuxserver/webtop project produces Docker containers of various Linux distributions (Ubuntu, Fedora, Arch, and Alpine) with different window managers (KDE, Mate i3, Openbox, IceWM, and XFCE) and for the x86-64, arm64, as well as the armhf architectures. After installing and launching one of these Docker containers, the desktop can be accessed immediately through the browser.

To enable the export of the graphical user interface and in order to allow access via the browser, the project integrates Apache Guacamole and the xrdp server software.

The configuration of the containers (e.g., specification of port numbers, UID, GID, timezone, etc.) is possible in the file docker-compose.yml. The containers' GitHub repositories are open to the public, the project is under active development and is also well documented. Using the prepared containers is trivial. Customizing the containers, from exporting the complete desktop to exporting individual applications, should be possible.

Similar Projects are docker-baseimage-gui, which includes the TigerVNC server, the noVNC proxy, and x11docker, which includes the x11vnc server and a xpra HTML5 client.

4.2.2 Ulteo Open Virtual Desktop

The Ulteo Open Virtual Desktop [56] is free software under the GNU General Public License (GPL) that is mainly written in C and is also a Debian-based Linux distribution of the same name. The latest version (v4.0.3) is from 2016. The included HTML5 client (based on Apache Guacamole) allows users to access the desktop via browser in addition to the native client. The source code can be found on GitHub, and installable packages are available elsewhere. Unfortunately, the vendor's web presence is no longer available, and documentation about the solution is hard to find.

After installing the software, users can log in via the browser and install new Linux and Windows hosts and applications via an administration page. After logging in via the browser, users can select from the installed applications and launch them individually.

Afterwards, they can be selected individually in the browser window and are opened on the desktop in the browser. The entire desktop can be accessed, including the window bar and all windows are movable. The whole Windows desktop is exported via RDP. Linux desktops are exported via VNC and in later versions using RDP [47]. Individual applications can be started in full-screen in the browser. In the so-called portal mode, an overview of the available applications can be displayed in the browser and enables users to run applications released by the administrator.

4.2.3 Kasm Workspaces

Kasm Technologies develops several free software solutions for operating system instances running inside containers, which can be accessed via the browser. In addition to that, the company also sells licenses for commercial versions.

Two of these solutions are Desktop Workspaces and Application Workspace. While Desktop Workspaces make an entire desktop in a Docker container accessible via the browser, an Application Workspace isolates individual Linux applications.

The company offers tools to deploy its solution on Amazon Web Services (AWS), Oracle Cloud, and DigitalOcean, or also to several Linux distributions like CentOS, Debian, and Ubuntu.

Kasm Technologies uses a self-developed, open-source VNC server and a HTML5 client, licensed under the GPL 2.0 license, as proxy software. An extensive selection of Docker containers with desktops or individual, isolated applications is available in a separate GitHub repository.

4.2.4 Running Windows VMs Using the QEMU/KVM Hypervisor

Windows VMs can run in Linux using the free QEMU/KVM hypervisor. Many setup instructions for specific Windows versions are available online.

A variety of frontends exist for administration of such QEMU/KVM-based virtual machines (and also of LXC containers). There is the Canonical-backed LXD project [41], virt-install via the libvirt abstraction layer [7], and Proxmox VE as a web interface capable of handling multi-node scenarios [22, 34]. A solution based on QEMU/KVM has the major advantage that Linux can be used as the server operating system.

4.2.5 Running Windows VMs Using the QEMU/KVM Hypervisor and Managing them Using Linux Docker Containers

Access via RDP or VNC to a Windows instance from within a Docker container is currently not easily feasible with publicly available tool chains. A hack that existed until 2018 has since been stopped by Microsoft. Installing Windows VMs using the KVM/QEMU hypervisor and the administration from a Docker container is one possible way to go. This is in general done by breaking the Docker isolation concept and by passing KVM control over to the container. Utilizing this approach, accessing the Windows desktop via RDP might still be possible. However, concerning the expected administrative effort, it is unclear whether such a solution makes sense.

4.2.6 Running Linux Docker Containers with Wine

Another solution for running Windows applications using only free software is made possible by using Wine, which provides a re-implementation of several essential Windows APIs. The progress of the Wine software development over the last two decades has been impressive, and the solution is often sufficiently compatible with many Windows applications. However, not all applications are guaranteed to work flawlessly.

Numerous projects exist that provide Docker containers that include a solution for web access. Table 1 contains an incomplete list of such projects.

4.2.7 Running Windows Server Containers

Windows Server Containers allow Windows-based applications to run in an isolated runtime environment. Analogous to Linux-based container virtualization, Microsoft offers an ecosystem for packaging, executing, and management of program packages.

Such program packages typically contain on one hand, the kernel of the host operating system and, on the other hand, the program logic introduced by the end-user within the guest container.

For depositing such program logic, initial base images are provided by Microsoft, which contain essential basic components in different stages of development:

- Windows base OS container image
  `https://hub.docker.com/_/microsoft-windows`
- Server base OS container image
  `https://hub.docker.com/_/microsoft-windows-server`
- Windows Server Core base OS container image
  `https://hub.docker.com/_/microsoft-windows-servercore`
- Nano Server Insider base image for containers
  `https://hub.docker.com/_/microsoft-windows-nanoserver-insider`

The deployed base images can then be incrementally extended by adding user-specific layers. Windows Server containers can thus be combined, efficiently transported, and also executed to represent lightweight and full-featured applications or services, similar to the approach taken with Linux-based containers.

For management, orchestration, and cloud integration, Microsoft essentially relies on the Microsoft Azure product family and related sub-technologies. Through services such as the Azure Container Registry, Windows Server Containers can often be usefully extended and seamlessly integrated into existing Windows infrastructures or integrated without significant configuration effort. This can be used advantageously in many cases. Still, using Windows Server Containers also creates further technical dependencies and relevant legal restrictions when inserting them into a modern DaaS.

According to vendor specifications, the native operation of Windows Server Containers must be provided from a supported and natively installed operating system (Windows Server 2016/2019/2022, Windows 10 Professional, Enterprise, and Education Editions). A further essential condition for native operation is the Hyper-V virtualization technology, which must be provided by the host operating system.

Additional conditions on the host system's hardware exist for the virtualized operation of Windows Server containers. For this purpose, a virtualized Hyper-V host must have at least 4 GB RAM, and the host CPU as well as the mainboard used must support the VT-x/AMD-V CPU extensions. In addition, the container host VM must have at least two virtual processors [15].

Apart from the technical conditions described, the licensing model specifies additional essential requirements for using the Windows server containers. In principle, every native Windows installation and virtualized Windows instance must be treated according to the applicable licensing model.

Prices, fees, and the scope of services are defined by the CAL (Client Access License) used, which is provided in the form of User CALs or Device CALs. A User CAL permits the use of several end devices by one user, and a Device CAL allows the use of one device by several users.

In addition, additional CALs may be required as soon as specific Microsoft-specific components are used. For

Table 1: Linux docker container projects with a VNC or RDP service and a HTML client

Project	Wine included	VNC Server included	RDP Server included	Proxy or HTML Client included	Active Project
wine-x11-novnc-docker	yes	x11vnc	no	noVNC	yes
Docker-WineHQ-VNC	yes	TightVNC	no	no	no
boggart/docker-wine-vnc	yes	TigerVNC	no	no	no
docker-nvidia-egl-desktop	yes	x11vnc	no	noVNC	yes
scottyhardy/docker-wine	yes	no	xrdp	no	no

example, using Docker base images may require other license terms. Furthermore, an additional RDS CAL is required if multiple RDP sessions are being utilized.

4.2.8 Running Linux Containers in Windows

Linux containers can run in Windows, allowing Linux-based applications to be executed in an isolated runtime environment. Analogous to Windows server containers, Docker offers an independent ecosystem for the encapsulation, execution and management of programs.

The program logic deposited by the user can be executed within the guest container and uses the host operating system's kernel. Program packages can be created and exchanged by using specific configuration files (e.g. Dockerfiles). A large variety of command line tools (e.g., "docker" or "docker-compose") published by the vendor or other third parties can be used to execute such packages. Access to numerous applications and services also exists through provided base images on public registries such as Docker Hub.

For these purposes the Windows Subsystem for Linux (WSL) in version 1 or 2 can be used when running Windows. WSL or WSL2 allows users to initially operate a GNU/Linux environment directly under the hood of Windows. In an additional step, it is possible to use standard Linux-based containers to execute the user program in an isolated way.

An advantage of this solution is that a large number of available applications are already available through the use of public registries. Moreover, this method causes several technical dependencies, like running a proprietary host operating system. Another major prerequisite for running the WSL2 essential components and the minimum hardware requirements is the presence of the Hyper-V virtualization technology. Supported host operating systems are currently only Windows Server (2016, 2019, and 2022) and Windows 10/11 (Enterprise, Pro, and Education). In virtualized operation mode, Hyper-V must be run on the host side and passed into the guest virtual machine. These guest virtual machines must then also contain a supported operating system fulfilling all minimum hardware properties. In addition, the CPU and mainboard used must support the VT-x/AMD-V CPU extensions.

4.2.9 Conclusion

Table 2 provides a summary of the projects and solutions described in this section and compares their criteria relevant to this work.

5 Identification and Analysis of Possible Components for Developing a Novel DaaS

The technical basis for enabling users to deploy Linux and Windows applications in a DaaS can be, among other things, application virtualization or container virtualization solutions (e.g., Docker, OpenVZ, or LXC). In this way, it is possible to isolate applications from each other in order to improve security and stability. For running Windows applications, choosing a solution that uses the Wine API emulator or something similar may be required.

Server-side execution of a Linux or Windows application in a container or virtual machine does not yet enable remote access via a browser. However, various options are investigated in this paper and with respect to their technical and non-functional characteristics. The transfer and display of the graphical output of individual applications that the users upload are possible, for example, using the VNC, RDP, and SSH+X11 protocols. This document analyzes and evaluates numerous free and commercial tools for this purpose, considering their suitability in a modern DaaS.

5.1 Server-Side Services for Remote Access to the UI of Linux or Windows Applications

Numerous server services for desktop export are available for Linux and Windows. As a rule, however, the solutions are designed entirely for exporting the entire desktop and not just individual windows or applications. Therefore, this section focuses on solutions capable of exporting individual applications.

Table 2: DaaS projects and solutions container virtualization

Project, Solution, or Product	Category	Virtualization Technology included	Support for non-modified... Linux-Applications	Windows-Applications	Free Software License
linuxserver/webtop	Webtop with Guacamole	Container (Docker)	yes	no	yes
Kasm Workspaces	Webtop and isolated applications with KasmVNC	Container (Docker)	yes	no	yes
Windows VMs with QEMU/KVM hypervisor	Installation of Windows VMs on a hypervisor in Linux	VM (QEMU/KVM)	no	yes	yes (except the Windows OS)
Linux Container with Wine	Webtop with Wine, VNC or RDP server and Guacamole	Container (Docker)	yes	yes (partly)	yes
Windows Server Container	Installation of Windows Server Containers	VM (Hyper-V)	yes (partly)	yes	no
Linux Container in Windows	Installation of Linux Containers	Container (WSL2)	yes	no	yes

5.1.1 NoMachine (NX), FreeNX und Neatx

Theoretically, the technology [35] from NoMachine could be used for a new DaaS project. The solution enables the display of individual Windows applications in seamless mode. However, the solution has been proprietary since version 4.0 of 2010 and would, in most scenarios, be too expensive for a DaaS. Therefore, NoMachine is only free of charge for private users. Up to and including version 3, the solution was called NoMachine NX and was free software under the GPL license. Unfortunately, this version is no longer maintained by the manufacturer. The last free source code under the GPL license is today available as the project FreeNX.

An open-source version of FreeNX formerly developed by Google, also under the GPLv2 license, called Neatx, has not been maintained for more than a decade and is severely outdated at the current time. Whether it would be possible at all to transfer individual windows with NoMachine NX, FreeNX, and Neatx is unknown and irrelevant regarding the costs or lack of topicality of the open-source solutions.

5.1.2 OpenSSH

Another possible solution is Secure Shell (SSH) integration (using the SSH client's command line options -X and -C to transfer the graphical user interface and use compression). SSH servers are installed by default on Linux and other UNIX-like operating systems. Usually, this is the OpenSSH [45] solution, which is free software under the BSD license and has been actively developed for many years.

However, the -X option only connects to the client's X11 server. No rasterized screen is transmitted, but commands for drawing the graphical user interface. Thus, another container with the X server would be necessary, as well as a screen transfer technology like VNC to transfer the frame buffer to the end user.

5.1.3 SharedAppVNC

The solution SharedAppVNC [50] is available for Linux, Windows, and MacOS. The software enables the export of individual windows and the entire desktop. Unfortunately, the project has not been developed since 2006. Therefore, whether using modern operating systems and a new DaaS solution is meaningful and possible must be doubted. The software is open source, but no software license is specified.

5.1.4 TightVNC

The free software TightVNC Server implements a VNC server for Windows that is licensed under GPL 2.0. The software is actively developed further by the manufacturer GlavSoft and distributed commercially. While older versions of TightVNC were also offered for Linux and macOS, the manufacturer's focus has been primarily on Windows operating systems for several years. The software implements a Single Application Sharing Mode, which can be useful in a DaaS.

5.1.5 TigerVNC (Xvnc and x0vncserver)

The free software TigerVNC Server implements a VNC server for Linux and Windows licensed under GPL 2.0. The software is being actively developed further. The current version (v1.12.0) is from 2021.

The project contains two different VNC server implementations: `Xvnc` and `x0vncserver`.

With `Xvnc`, as with any typical VNC server, a virtual display is started for each new connection. The server service is often started by the wrapper script vncserver. No command line parameter for exporting only individual applications is apparent in `Xvnc`'s documentation.

The `x0vncserver` server service allows direct control over a physical X session. Thus, the actual image is displayed, and no virtual display is started. Also, in the man page of `x0vncserver` no command line parameter for the export of only single applications is recognizable. The only way to limit the desktop area to be transferred is the geometry and video area parameters. However, the area to be transferred would have to be known in advance for each window and specified when the server service is started, which is impractical and possibly unreliable.

5.1.6 x11vnc

The free software x11vnc allows the export of complete desktops or individual applications from Linux desktops. The software is licensed under GPL2 and is being actively developed. Installation, configuration, and operation of x11vnc are simple and well-documented.

5.1.7 xrdp

The xrdp server is free software (Apache 2.0 license) for Linux and UNIX-like operating systems. The software has been in development for almost two decades and is being actively developed. By default, the daemon exports the complete desktop. However, it also seems possible to export individual applications.

5.1.8 Windows RDP Server

The default Windows RDP server can export individual applications via RDP. To do that the Active Remote Desktop has to be enabled at first. For any hosted application a registry key under `/HKLM/SOFTWARE/NT/CurrentVersion/Server/TSAppAllowList/Applications` has to be manipulated in order to map a certain key name (any application name) to an (absolute) path of an executable binary.

5.1.9 Conclusion

Table 3 summarizes the projects and solutions presented in this section and compares their criteria relevant for our proposed DaaS design.

The free solutions x11vnc for the VNC protocol, Xrdp for the RDP protocol, and OpenSSH for the SSH protocol are all fully suitable for remote access to graphical applications in Linux instances. For technological reasons, it is irrelevant whether these Linux instances run directly on physical hardware or in virtual machines or whether they are containers.

The VNC servers `Xvnc` and `x0vncserver` of the free solution TigerVNC for Linux does not offer the possibility to transfer only single windows or applications. They are, therefore, not suited for a new DaaS project.

The solution of NoMachine does not come into consideration for reasons of the license and because of the costs which can be expected for a DaaS. Furthermore, the free software projects FreeNX and Neatx, derived from the last free source code about 10 to 15 years ago, are outdated and, therefore, not a recommendable solution.

For Windows, the open-source TightVNC server for the VNC protocol and the integrated screen sharing based on the RDP protocol exist. This is being actively developed further, well documented, and offers a solution independent of Microsoft's on-board solutions.

5.2 Proxy Solutions for Displaying Graphical Output in the Browser

The free solutions Apache Guacamole and noVNC work as a proxy between a VNC service on the server side and the browser. The proxy can also run on the server side or at another location between the server and the clients and eliminates the need to install additional software on the client in order to interact with the remote programs. In the case of a new DaaS project, installing additional client software would be counterproductive because it contradicts seamless integration with DaaS and unnecessarily complicates working with a DaaS from the user's perspective and access, thus contradicting the design principles of a modern DaaS solution.

5.2.1 Guacamole

Apache Guacamole [19, 24] is a remote desktop gateway installed exclusively on the server side. Apart from a browser, no other software is required on the client. The software supports VNC, RDP, and SSH. Guacamole is free software under the Apache License 2.0 and is under active development. In addition to extensive

Table 3: Services for remote access to individual Linux or Windows Applications

Project, Solution, or Product	Category	Supported Protocol	Export of single Window	Supported OS	Software License	Active Project	Useful for a DaaS
NoMachine	Terminal server	NX	yes	Linux	proprietary	yes	no
FreeNX	Terminal server	NX v3.x	yes	Linux	GPL2	no	no
Neatx	Terminal server	NX v3.x	yes	Linux	GPL2	no	no
OpenSSH	SSH server	SSH	yes	Linux	BSD	yes	limited
SharedAppVNC	VNC server	VNC	yes	Linux, Windows	GPL2	no	no
TightVNC	VNC server	VNC	yes	Windows	GPL2	yes	yes
TigerVNC	VNC server	VNC	no	Linux, Windows	GPL2	yes	no
x11vnc	VNC server	VNC	yes	Linux	GPL2	yes	yes
Xrdp	RDP server	RDP	yes	Linux	Apache 2.0	yes	yes
RDP server in Windows	RDP server	RDP	yes	Windows	proprietary	yes	yes

documentation for users and developers alike, the project provides an easy way to install via Docker containers. Copy and paste of text between local and remote systems is possible.

The software contains two components:

- Guacamole, the web frontend, is implemented in JavaScript and interacts with the user's browser.
- guacd, the proxy that connects to desktops via VNC, RDP, or SSH, handles communication with the frontend.

5.2.2 KasmVNC

Kasm Technologies develops an open-source VNC server, incl. a web server [39]. The solution also includes an HTML5 client. The software components are licensed under the GPL 2.0 license and are being actively developed further. The server is accessed exclusively via the browser. The protocol used is a modification of VNC and is incompatible with otherwise ordinary VNC viewers. The company provides Debian/Ubuntu and CentOS installation packages and extensive documentation on the GitHub pages and its corporate website.

5.2.3 noVNC

The software noVNC [11, 32] is free software developed in JavaScript to display a remote computer's screen content in a remote client's browser. Interaction via mouse and keyboard from the client to the remote desktop is possible. The software is under active development and is used productively by numerous projects (e.g., OpenStack and OpenNebula). noVNC requires a VNC server with support for WebSockets. If the VNC server does not implement WebSockets, an appropriate proxy such as websockify must be installed on the server. After installing noVNC and running the software, it starts a daemon on the client that is accessible via its port number and connects to the port number of the VNC server. The browser on the client calls the noVNC daemon via its port number and can thus access the server's desktop via VNC. For simplified installation, the project provides snap packages, among other things.

5.2.4 Spice Web Client (eyeOS)

The company eyeOS in Barcelona developed the so-called Spice Web Client around 2015/2016. This is implemented in HTML5 and JavaScript and is free software (MIT License). The software has not been further developed since 2016. Spice is a remote desktop protocol whose development was driven by RedHat for a while. The SPICE protocol, as well as client and server implementations, are free software [13, 27]. A SPICE server for Windows probably does not exist. Advantages of the SPICE protocol are that USB devices can be passed very well from client to server. Also, copy and paste between client and server should work smoothly. Reference implementations of SPICE include an HTML5 client. This was created around the same time as the Spice Web Client from eyeOS but reportedly has significantly worse performance data.

5.2.5 Xpra

Another option for transferring the display of individual Linux applications is to use the Xpra (X Persistent Remote Applications) solution. Xpra is a ”screen for X11” with which graphical programs can be used in the network. In addition, the connection can be interrupted and resumed later from the same or another computer without terminating the respective program. With xpra-html5, the HTML5 client for Xpra, access to individual Linux applications via the browser is possible. Xpra is free software and is licensed under GPL2. The project is active.

5.2.6 Conclusion

Table 4 summarizes the solutions presented in this section and compares their criteria relevant to a new DaaS project.

Apache Guacamole is published under a open-source license and a powerful tool for using applications and desktops via the VNC, RDP, and SSH protocols Other established and free tools with a comparable but smaller range of functions are noVNC and Xpra.

A complete ecosystem of individually or completely usable solutions is Kasm, which offers a powerful and free-software VNC server, including an HTML5 client. In addition, the ecosystem of Kasm can stream complete desktops in Docker containers and individual, isolated applications in Docker containers.

The SPICE Web Client from eyeOS has not been developed further for several years. Therefore, it is unlikely that it is superior to current software solutions like Guacamole with VNC or RDP. Furthermore, it would not help to integrate Windows applications since there is no SPICE server for Windows.

The RemoteApp for Windows is a solution from Microsoft that is included in the server versions of Microsoft Windows. This tool is generally also capable of making individual applications accessible via the browser.

5.3 Tools for Orchestration, Monitoring and Management of Distributed Machines and Containers

Since a modern DaaS infrastructure will not only comprise a single virtual machine or container but a large number of such distributed across multiple physical nodes, powerful tools for orchestration, monitoring, and management of the distributed machines and containers are required. This section presents a selection of available tools.

5.3.1 Kubernetes (K8s)

Kubernetes (K8s) is a free software (Apache 2.0 license) for automated deployment, scaling, and management of containerized applications [8]. The software is actively developed and used in projects worldwide. It is considered very sophisticated but complex to use. All tasks of a Kubernetes cluster are run redundantly on multiple servers.

Kubernetes can interact with different container runtimes (Docker Engine, containerd, CRI-O API). Furthermore, using the daemon LXE, Kubernetes can also interact with LXC/LXD resources. With KubeVirt, a virtual machine management add-on, Kubernetes can also manage virtual machines based on the QEMU/KVM hypervisor [4, 46, 57].

5.3.2 LXD Dashboard

LXD Dashboard is free software (GPL3 license) that implements a web interface to manage instances of the Linux container daemon LXD in multi-node scenarios. The project is under active development and the driving force behind the product is the company LXDWARE. The company also provides the LXD Dashboard as a Docker container. Operation in an LXC container is also possible without any problems. The solution is developed in PHP. The installation of the software is well documented. The software makes a mature and stable impression. The range of functions differs from solutions like Proxmox, but it is in general very lightweight.

5.3.3 LXDMosaic

LXDMosaic is another free software (GPL3 license) that implements a web interface to manage instances of the Linux container daemon LXD in multi-node scenarios. The project is more or less actively developed. The current version is v0.16.0, and installation and operation are relatively simple and well-documented. The project appears in every respect to be a hobby project driven by a single person (”turtle0x1”).

5.3.4 Proxmox Virtual Environment (PVE)

Proxmox VE is an open-source server virtualization platform that combines KVM and container-based virtualization and manages virtual machines, containers, storage, virtual networks, and high-availability clusters from the central management interface [3, 26].

Proxmox works only with VMs based on QEMU/KVM and containers based on LXC. The project is under active development and is free software

Table 4: Proxy solutions for displaying graphical output in the Browser

Project, Solution, or Product	Category	Supported Protocols	Supported OS	Software License	Active Project	Useful for a DaaS
Apache Guacamole	Gateway	VNC, RDP, SSH	Linux	Apache 2.0	yes	yes
KasmVNC	VNC server and HTML5 client	Modified VNC	Linux	GPL	yes	yes
noVNC	Gateway	VNC	Linux	MPL 2.0	yes	yes
Spice Web Client from eyeOS	HTML5 client	Spice	Linux	MIT	no	no
Xpra	HTML5 client	Xvfb	Linux	MPL 2.0	yes	yes

under the AGPLv3 license. The current version, v8.0, is based on Debian GNU/Linux 12.

The software is used in many successful projects worldwide and is considered very sophisticated and feature-rich. The company behind the development, Proxmox Server Solutions GmbH, sells support licenses and other software.

5.3.5 Rancher

Rancher is a solution to manage multiple Kubernetes clusters in different cloud environments. Rancher is particularly useful for multi-cloud scenarios (simultaneous use of multiple public cloud service offerings and local resources). For example, a typical deployment scenario is DevOps teams developing locally in RKE1 or RKE2 (Rancher's next-generation Kubernetes distribution) and then "deploying" to public cloud resources (e.g., Microsoft Azure AKS, Amazon EKS, and Google GKE). Rancher supports several Linux distributions for local installation, including SLES, CentOS, RHEL, Oracle Linux, and Ubuntu. However, Debian is not included and only focused on Kubernetes (K8s).

Rancher's source code is actively maintained and available as free software under the Apache 2.0 license.

5.3.6 Conclusion

Table 5 summarizes the solutions presented in this section and compares the criteria relevant to our project.

Kubernetes and Proxmox are the most recommended solutions in terms of fit concerning functionality and reliability, as well as the quality of the solution and existing documentation. The solutions LXD dashboard and LXDMosaic have more of a hobbyist character. Using Rancher in a DaaS does not make sense currently, as it is a solution to manage multiple Kubernetes clusters in different environments. However, this needs to be provided in the architecture concept of our new DaaS project.

6 Recommend Architecture of the DaaS

Numerous combinations of the software solutions being presented in this paper are conceivable and possible. An architecture for a new and modern DaaS requires several layers of functions implemented by different software solutions. Above the operating system, virtualization solutions must exist for all viable platforms. Therefore, Hypervisor solutions for operating virtual machines and container virtualization come into question. Of the tools for orchestrating, monitoring, and managing distributed machines or containers, Proxmox VE and Kubernetes (K8s) are the most recommended.

Kubernetes is the right choice when multiple sites are to be linked and when local resources and resources at a public cloud service provider are used simultaneously. However, if this is not the case, Proxmox is the solution with the most advantages, as it is easy to install and operate, and with its support for VMs based on KVM/QEMU, it can be used very flexibly. However, Proxmox does not support Docker, but only LXC containers out of the box.

A modern DaaS must be able to run Linux and Windows applications. From our perspective, a container solution with Linux containers is the best choice for running Linux applications, as the applications run isolated, and this approach generates little overhead.

The API of LXC is, in comparison to the Docker API, not very feature-rich and poorly documented. Therefore, Docker is most likely the best solution as a container solution for a DaaS for running Linux containers. Running Docker in Linux VMs is a solution that is well-established in practice and causes only little overhead.

The stability and range of functions of the existing extensions for Proxmox in order to manage Docker containers need to be clarified and are doubtful. Controlling Docker via the Docker API is sufficient for most DaaS usage scenarios. However, using another container orchestration solutions must be considered in

Table 5: Proxy solutions for displaying graphical output in the Browser

Project, Solution, or Product	Supports Virtual Machines	Supports Containers	Software License	Active	Useful for a DaaS
Kubernetes (K8s)	yes (with KubeVirt extension also QEMU/KVM)	yes (e.g. Docker. With LXE Daemon also LXC/LXD)	Apache License 2.0	yes	yes
LXD dashboard	yes (LXD)	yes (LXD)	AGPLv3	yes	no
LXDMosaic	yes (LXD)	yes (LXD)	GPL 3.0	yes	no
Proxmox VE	yes (QEMU/KVM)	yes (LXC)	AGPLv3	yes	yes
Rancher	no	Kubernetes (K8s)	Apache License 2.0	yes	no

a multi-node scenario. For example, Kubernetes or Portainer would be suitable, although it still needs to be determined whether Portainer would help us. Any of these solutions would make sense to run in a VM.

Running Windows applications is often possible in Linux containers with Wine. However, this approach is unfeasible for some Windows applications, but for those it supports, it is a simple and resource-efficient solution. For Windows applications unsupported by Wine, virtual machines and a fully featured Windows operating system are mandatory since Windows containers do not export graphical interfaces.

To export the graphical user interface of the Linux and Windows applications, a protocol such as VNC or RDP is required. As shown in this document, there are implementations for both protocols that are stable, reliable, and free software. For Linux, these are x11vnc and Xrdp. For Windows, these are TightVNC and the Windows internal RDP server. Ideally, a modern DaaS solution only exports the respective application, rather than a complete desktop, which is generally supported by the solutions mentioned above.

In addition to running Linux and Windows applications and exporting their graphical output, a modern DaaS must allow interaction via a browser. For that purpose, software that mediates between VNC or RDP and the browser is required. Hence, a minimal VNC client such as noVNC or a full-featured Desktop-Gateway such as Apache Guacamole is necessary. While noVNC can only interact with VNC, Apache Guacamole can work with VNC and RDP. Both solutions are conceivable and sufficiently stable, whereby noVNC is the more lightweight solution.

Figure 1 shows an architecture that can run unmodified Linux and Windows applications in Linux containers (Linux and Windows applications compatible with Wine) and virtual machines (Windows applications incompatible with Wine). The architecture is designed to export only the application's graphical user interface, and interactions are purely made via browser. No additional client software on the user site is required to interact with the DaaS and its applications.

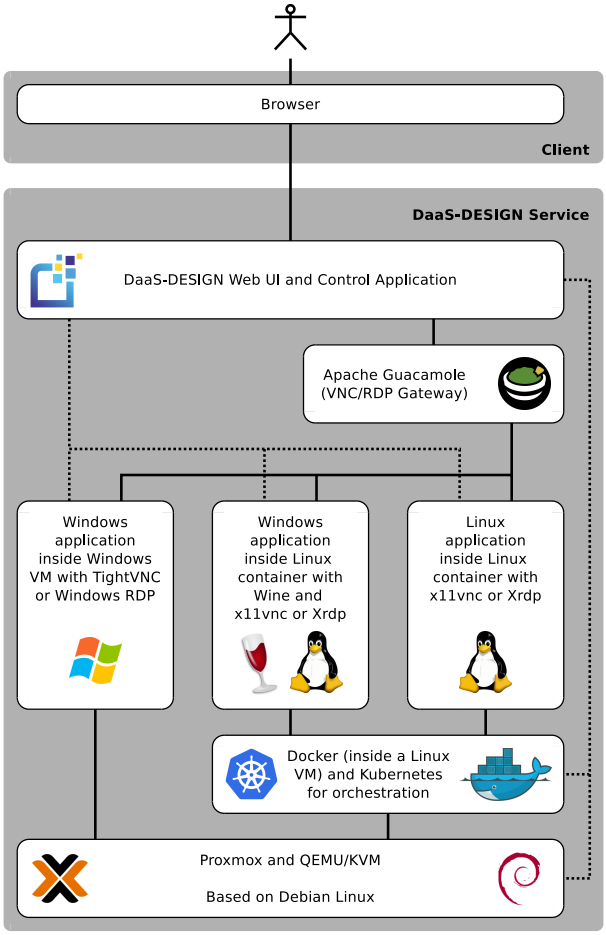

Figure 1: Recommend architecture for a novel, feature-rich and innovative DaaS

A novel DaaS, based on the described architecture in Figure 1, that has the potential to close the Gap between Web Applications and Desktop Applications is developed and implemented in the project DESIGN [18].

6.1 Technical Challenges and Solutions in Developing the DaaS Solution Proposed

Developing and implementing a novel DaaS, as the solution propagated in this work, includes multiple technical challenges. A selection of the most urgent challenges and solutions is given in this section.

6.1.1 User Interaction

In general, providing a graphical interface for user interaction is a challenge on its own. For most basic use cases, the problem can sufficiently be solved by using one of the frame buffer protocols described earlier. Nevertheless, for certain edge cases and particularly within the described DaaS scenario, such basic solutions might not always provide a satisfying and fully transparent user experience.

Most VNC- and noVNC-based implementations benefit from the lightweight and efficient protocol design, which allows Proxmox to offer out-of-the-box VNC support for all platforms and independent of any user configuration. RDP-based solutions can benefit from it's rich feature support, which provides user session control and several other useful optimizations.

For our particular DaaS scenario, one of the most challenging aspects is that the graphical export of individual applications is only implemented by some server solutions (see section 5) and is not always trivial in practice since individual applications might open multiple windows or resize its windows depending on a certain application state.

RDP generally supports that but is only available in an already installed and configured Windows environment. Otherwise, VNC/noVNC solutions can also be used in these scenarios but most of the implementations offer only limited support for individual applications.

Therefore, for contexts involving pre-installed Windows environments the RDP protocol might be the superior solution, while for all other contexts, VNC/noVNC-based solutions might still be preferable. As a result, we decided to base our solution on Apache Guacamole which can work with both protocol families.

Implementing support for both protocols in both operating systems offers much more flexibility at first sight, but it also increases the complexity of the DaaS solution. Focusing on just a single protocol may become a limiting factor.

For our web-based services, we therefore implemented a dedicated proxy component exchanging websocket messages between the guacamole daemon (guacd) and a JavaScript client utilized by the web user. By default, the guacd then interacts with the specified platform in VNC or RDP but can also be extended for any other protocol if needed. This flexible and scalable approach offers a still acceptable user experience and performance while also minimizing the integration effort. Additionally, the proxy approach allows to react to certain protocol-level events, such as, for example, screen resize messages.

6.1.2 Ressource Requirements

From a user perspective, the time to deploy and make applications accessible may not take too long. Otherwise, the users might consider the DaaS system as broken or unresponsive.

From a provider perspective, the resource requirements when running multiple Linux Containers and Windows virtual machines are generally hard to predict and, thus, the resource consumption of the entire DaaS deployment is also hard to predict.

As our DaaS scenario depends on several technical layers, the particular challenge imposed for our approach is, therefore, essentially to predict and control the performance of a set of inter-operating services which provides a specified application with a satisfying user experience. In that sense, each service's performance aspects, especially their combination, are crucial for the success of the whole project.

Based on our experiments, the required time to deploy applications depends on the hardware resources of the underlying servers, the virtualization solution used (virtual machine or container), the solution to provide the graphical output, the operating system, and last but not least, the application and its size itself. Different ways to deploy user applications as well as the required time for different applications must be analyzed, and ways to optimize the necessary steps must be investigated.

Therefore, our solution offers three main use cases that enable users to choose the optimal hosting strategy for their specific problem to solve (see Figure 1). Generally, it can be said that containers with and without support for the Wine software are usually more lightweight in terms of memory or CPU consumption and are, therefore, the preferable choice performance-wise. Otherwise, some Windows applications are only available in fully virtualized environments and therefore, a virtual machine is always possible as a last resort.

6.1.3 User Isolation

The applications running inside containers and virtual machines are kept in an isolated way, but must still be able to access shared data. The DaaS must, therefore, implement a storage solution that has a high level of reliability and scales well in multi-node environments.

Proxmox contains the distributed file system Ceph [52, 53], which is considered sufficiently stable and can scale well [23, 58], but integrating access to Ceph from containers and virtual machines of different operating systems in an automated and secure way might be a challenging task. In order to do that a solution has to distinguish at least between public, user-related and shared data on an architectural level in order to prevent unauthorized access or data loss.

Installation of new containers or VM's can then be done either by using a set of recommended packages from global or organizational repositories, or also by using user-specific installers from private storages. For that purpose we implemented a dedicated Python component running within any hosted container or virtual machine being able to run applications independently from the actual operating system. With this approach most standard applications might be controlled but as this is heavily depending on the application itself, and therefore customization might still be necessary for non-standard usecases.

7 Conclusions and Next Steps

In this paper, we propagate an architecture for a novel and feature-rich DaaS that supports running unmodified Linux and Windows applications and we examined various solutions and existing software tools for all levels of the architecture. The solution can export entire desktops or single applications in a favorable manner. Interaction of the users is done just via a browser, allowing interaction from all sorts of devices and host operating systems. Furthermore, almost all software components are free software (except the Windows operating system in the Windows VMs).

Further next steps are implementing the DaaS by combining the selected open source software components and developing the required software to interact with all elements of the DaaS. Developing a DaaS application that can create, control, monitor, and remove containers via Docker API or VMs via the Proxmox API is essential. In addition, the DaaS application must be able to automatically equip new instances with the desired application, a VNC or RDP daemon, and their configuration.

For future works, it will be crucial as well to measure and evaluate performance characteristics of a DaaS reference implementation. Such performance testing should include different user behavior, a variety of plausible applications, as well as realistic deployment and utilization scenarios. Therefore, suitable test procedures and metrics have to be found, defined and evaluated.

Acknowledgements

This work was funded by the Federal Ministry for Economic Affairs and Climate Action ('*Bundesministerium für Wirtschaft und Klimaschutz*') in the framework of the central innovation programme for small and medium-sized enterprises ('*Zentrales Innovationsprogramm Mittelstand*').

We thank our project partners from Nuromedia GmbH for their support. We especially thank Björn Goetschke, Mike Ludemann, Dario Savella, Holger Sprengel, and Rahul Tomar. We would also like to thank Lukas Atkinson for his review of a draft version of this paper.

References

[1] R. Afreen, "Bring your own device (BYOD) in higher education: Opportunities and challenges," *International Journal of Emerging Trends & Technology in Computer Science*, vol. 3, no. 1, pp. 233–236, 2014.

[2] M. Afrin, J. Jin, A. Rahman, A. Rahman, J. Wan, and E. Hossain, "Resource Allocation and Service Provisioning in Multi-Agent Cloud Robotics: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 842–870, 2021.

[3] S. A. Algarni, M. R. Ikbal, R. Alroobaea, A. S. Ghiduk, and F. Nadeem, "Performance evaluation of xen, kvm, and proxmox hypervisors," *International Journal of Open Source Software and Processes (IJOSSP)*, vol. 9, no. 2, pp. 39–54, 2018.

[4] M. Amaral, "Kubevirt scale test by creating 400 vmis on a single node," in *Free and Open source Software Developers' European Meeting*, 2022.

[5] F. Bellard, "QEMU, a fast and portable dynamic translator," in *USENIX annual technical conference, FREENIX Track*, vol. 41. California, USA, 2005, p. 46.

[6] A. Binu and G. S. Kumar, "Virtualization techniques: a methodical review of XEN and KVM," in *Advances in Computing and Communications: First International Conference, ACC 2011, Kochi, India, July 22-24, 2011. Proceedings, Part I 1*. Springer, 2011, pp. 399–410.

[7] M. Bolte, M. Sievers, G. Birkenheuer, O. Nieh örster, and A. Brinkmann, "Non-intrusive virtualization management using libvirt," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. IEEE, 2010, pp. 574–579.

[8] B. Burns, J. Beda, K. Hightower, and L. Evenson, *Kubernetes: Up and Running*. O'Reilly Media, Inc, 2022.

[9] N. Burns-Sardone, "Making the case for BYOD instruction in teacher education," *Issues in Informing Science and Information Technology*, vol. 11, no. 1, pp. 192–200, 2014.

[10] A. Celesti, D. Mulfari, M. Fazio, M. Villari, and A. Puliafito, "Improving desktop as a service in openstack," in *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2016, pp. 281–288.

[11] L. Chen, W. Huang, A. Sui, D. Chen, and C. Sun, "The online education platform using Proxmox and noVNC technology based on Laravel framework," in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*. IEEE, 2017, pp. 487–491.

[12] S. Chen, T. Zhang, and W. Shi, "Fog Computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 4–6, 2017.

[13] L. Cheng, "Research on mechanism optimization of usb redirection transmission based on spice protocol [j]," *Computer Science and Application*, vol. 10, no. 6, pp. 1166–1179, 2020.

[14] Collabora. Collabora Online. [Online]. Available: https://www.collaboraoffice.com/collabora-online/

[15] M. Corporation. Windows 10 hyper-v system requirements. [Online]. Available: https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/hyper-v-requirements

[16] L. Deboosere, B. Vankeirsbilck, P. Simoens, F. De Turck, B. Dhoedt, and P. Demeester, "Cloud-based desktop services for thin clients," *IEEE Internet Computing*, vol. 16, no. 6, pp. 60–67, 2011.

[17] S. Dernbecher, R. Beck, and M. Toenker, "Cloudifying Desktops-A Taxonomy for Desktop Virtualization," in *AMCIS*, 2013.

[18] DESIGN project. Design. [Online]. Available: https://www.daas-design.de

[19] S. García, A. Gallardo, D. F. Larios, E. Personal, J. M. Mora-Merchán, and A. Parejo, "Remote Lab Access: A Powerful Tool Beyond the Pandemic," in *2022 Congreso de Tecnología, Aprendizaje y Enseñanza de La Electrónica (XV Technologies Applied to Electronics Teaching Conference)*. IEEE, 2022, pp. 1–5.

[20] A. Garmpis and N. Gouvatsos, "Design and development of webubu: An innovating web-based instruction tool for linux os courses," *Computer Applications in Engineering Education*, vol. 24, no. 2, pp. 313–319, 2016.

[21] J. Gedeon, F. Brandherm, R. Egert, T. Grube, and M. Mühlhäuser, "What the Fog? Edge Computing Revisited: Promises, Applications and Future Challenges," *IEEE Access*, vol. 7, pp. 152 847–152 878, 2019.

[22] R. Goldman, *Learning Proxmox VE*. Packt Publishing Ltd, 2016.

[23] D. Gudu, M. Hardt, and A. Streit, "Evaluating the performance and scalability of the ceph distributed storage system," in *2014 IEEE International Conference on Big Data (Big Data)*. IEEE, 2014, pp. 177–182.

[24] I. Hassan, "Levereging Apache Guacamole, Linux LXD and Docker Containers to Deliver a Secure Online Lab for a Large Cybersecurity Course," in *2022 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2022, pp. 1–9.

[25] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the Linux Virtual Machine Monitor," in *Proceedings of the Linux symposium*, vol. 1, no. 8. Dttawa, Dntorio, Canada, 2007, pp. 225–230.

[26] A. Kovari and P. Dukan, "KVM & OpenVZ virtualization based IaaS open source cloud virtualization platforms: OpenNode, Proxmox VE," in *2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics*. IEEE, 2012, pp. 335–339.

[27] Y. Lan and H. Xu, "Research on technology of desktop virtualization based on spice protocol and its improvement solutions," *Frontiers of Computer Science*, vol. 8, pp. 885–892, 2014.

[28] K. Liu and L.-j. Dong, "Research on cloud data storage technology and its architecture implementation," *Procedia Engineering*, vol. 29, pp. 133–137, 2012.

[29] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource Scheduling in Edge Computing: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021.

[30] E. Magana, I. Sesma, D. Morato, and M. Izal, "Remote access protocols for desktop-as-a-service solutions," *PloS one*, vol. 14, no. 1, p. e0207512, 2019.

[31] P. Mell, T. Grance *et al.*, "The NIST Definition of Cloud Computing," 2011.

[32] D. Mulfari, A. Celesti, M. Villari, and A. Puliafito, "Using virtualization and novnc to support assistive technology in cloud computing," in *2014 IEEE 3rd Symposium on Network Cloud Computing and Applications (ncca 2014)*. IEEE, 2014, pp. 125–132.

[33] P. H. Nakhai and N. B. Anuar, "Performance evaluation of virtual desktop operating systems in virtual desktop infrastructure," in *2017 IEEE Conference on Application, Information and Network Security (AINS)*. IEEE, 2017, pp. 105–110.

[34] V. Oleksiuk and O. Oleksiuk, "The practice of developing the academic cloud using the proxmox ve platform," *Educational Technology Quarterly*, vol. 2021, no. 4, pp. 605–616, 2021.

[35] G. F. Pinzari, "Introduction to NX technology," Technical report, Nomachine Inc, Tech. Rep., 2003.

[36] Prototype Fund, "SOFTWARE SPRINT (PROTOTYPE FUND) Konsolidierter Schlussbericht. AUSWAHLRUNDE MÄRZ 2018. Universe - Ein shared Webdesktop. Schlussbericht," Open Knowledge Foundation Deutschland, Tech. Rep., 2018. [Online]. Available: https://prototypefund.de/wp-content/uploads/2022/04/PTF_Abschlussberichte_Runde_4.pdf

[37] P. P. Ray, "An Introduction to Dew Computing: Definition, Concept and Implications," *IEEE Access*, vol. 6, pp. 723–737, 2017.

[38] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual network computing," *IEEE Internet Computing*, vol. 2, no. 1, pp. 33–38, 1998.

[39] S. A. Russo, S. Bertocco, C. Gheller, and G. Taffoni, "Rosetta: A container-centric science platform for resource-intensive, interactive data analysis," *Astronomy and Computing*, vol. 41, p. 100648, 2022.

[40] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[41] S. Senthil Kumaran, *Practical LXC and LXD: linux containers for virtualization and orchestration*. Springer, 2017.

[42] W. Shi and S. Dustdar, "The Promise of Edge Computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[43] A. Siani, "BYOD strategies in higher education: current knowledge, students' perspectives, and challenges," *New Directions in the Teaching of Physical Sciences*, vol. 12, no. 1, 2018.

[44] K. Skala, D. Davidovic, E. Afgan, I. Sovic, and Z. Sojat, "Scalable Distributed Computing Hierarchy: Cloud, Fog and Dew Computing," *Open Journal of Cloud Computing (OJCC)*, vol. 2, no. 1, pp. 16–24, 2015.

[45] M. Stahnke, *Pro OpenSSH*. Apress, 2005.

[46] S. Trakulmaiphol and K. Piromsopa, "An implementation of dns operator and network guideline for migrating virtual machine to kubevirt," in *2022 26th International Computer Science and Engineering Conference (ICSEC)*. IEEE, 2022, pp. 84–89.

[47] Ulteo SAS, "Ulteo open virtual desktop - protocol description," Ulteo SAS, Tech. Rep., 2008.

[48] UniverseyOS project. universeos prototype fund. [Online]. Available: https://prototypefund.de/project/universeos/

[49] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and Opportunities in Edge Computing," in *2016 IEEE international conference on smart cloud (SmartCloud)*. IEEE, 2016, pp. 20–26.

[50] G. Wallace and K. Li, "Virtually Shared Displays and User Input Devices," in *USENIX Annual Technical Conference*, vol. 7, 2007.

[51] Y. Wang, "Definition and Categorization of Dew Computing," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 1–7, 2016.

[52] S. A. Weil, "Ceph: reliable, scalable, and high-performance distributed storage," 2007.

[53] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th symposium on Operating systems design and implementation*, 2006, pp. 307–320.

[54] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog Computing: Platform and Applications," in *2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb)*. IEEE, 2015, pp. 73–78.

[55] A. Zakai, "Emscripten: an LLVM-to-JavaScript compiler," in *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, 2011, pp. 301–312.

[56] M. I. Zhaldak, V. M. Franchuk, and N. Franchuk, "Some applications of cloud technologies in mathematical calculations," in *Journal of Physics: Conference Series*, vol. 1840, no. 1. IOP Publishing, 2021, p. 012001.

[57] J. Zhang and M. Han, "Cloud native virtual computing cluster," in *2022 IEEE 8th International Conference on Cloud Computing and Intelligent Systems (CCIS)*. IEEE, 2022, pp. 273–277.

[58] X. Zhang, S. Gaddam, and A. Chronopoulos, "Ceph Distributed File System Benchmarks on an Openstack Cloud," in *2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. IEEE, 2015, pp. 113–120.

AUTHOR BIOGRAPHIES

Dr. Christian Baun is a Professor at the Faculty of Computer Science and Engineering of the Frankfurt University of Applied Sciences in Frankfurt am Main, Germany. He earned his Diploma degree in computer science in 2005 and his Master degree in 2006 from the Mannheim University of Applied Sciences. In 2011, he earned his Doctor degree from the University of Hamburg. He is the author of several books, articles, and research papers, e.g., in public and private cloud infrastructure and platform services, single-board computers, distributed computing, and distributed storage. His research interests includes operating systems, cloud services, distributed systems, and computer networks.

Johannes Bouché is a research assistant at the Faculty of Computer Science and Engineering of the Frankfurt University of Applied Sciences in Frankfurt am Main, Germany. He earned his Master degree in 2021 from the Frankfurt University of Applied Sciences. His research interests are computer networks, distributed systems, operating systems and information security.