

Kasm + Proxmox

Moderne VDI Lösung mit Autoscaling für Proxmox VE



Wir I(❤️)eben Open-Source.

Unser Engagement für Open-Source ist mehr als nur eine Philosophie – es ist das Herzstück unserer Arbeit und treibt unsere Innovation voran. Aus diesem Grund benutzen, fördern und entwickeln wir bei der creativ GmbH an Open-Source Lösungen jeglicher Art! Für unabhängige und freie Alternativen!

Über mich

Florian Paul Azim Hoberg

- Bekannt als **gyptazy** in der Open-Source & Tech-Community
- Head of Virtualization bei **credativ GmbH** (Mönchengladbach, Deutschland)
- **FOSS Contributor** (Freie und Open-Source-Software)
- Liebhaber von **FreeBSD**

Meine Persönlichen Projekte



Ansible

Aktiv als Upstream-Entwickler für Ansible-Module mit Fokus auf Enterprise-Linux-, BSD- und Virtualisierungsplattformen. Entwicklung und Pflege zahlreicher Module für RHEL, FreeBSD sowie Proxmox VE.



BoxyBSD / BoxedTux

BoxyBSD ist ein non-profit Projekt, das kostenlose virtuelle Server (VPS) mit BSD-basierten Betriebssystemen wie FreeBSD, OpenBSD und NetBSD für Lern- und Testzwecke bereitstellt.



ProxLB

ProxLB ist ein intelligenter Load-Balancer für Proxmox VE, der virtuelle Maschinen automatisch innerhalb eines Clusters verteilt, um Ressourcen optimal auszunutzen und die Gesamtperformance zu verbessern.



PegaProx

PegaProx ist eine Open-Source-Plattform für das zentrale Management von Proxmox VE-Umgebungen und ermöglicht die Verwaltung mehrerer Cluster über eine moderne, einheitliche Weboberfläche.

Was bedeutet VDI?

VIRTUAL DESKTOP INFRASTRUCTURE

Ein Desktop läuft nicht mehr zwingend auf dem physischen Endgerät des Benutzers. Stattdessen wird der Arbeitsplatz mitsamt des Betriebssystems, Anwendungen und Daten zentral auf einem Server betrieben. Das Endgerät dient nur noch als Anzeigemedium.

1

User Device

Browser oder Thin Client

2

Netzwerk

Verschlüsselte Übertragung

3

VDI Layer

Auth · Sessions · Desktop

4

Compute

CPU · RAM · Storage

Ziel: Den Arbeitsplatz vollständig vom physischen Endgerät entkoppeln.



Warum überhaupt VDI?

Klassischer Arbeitsplatz

Jedes Endgerät trägt sein eigenes Ökosystem:

- Lokales Betriebssystem
- Lokal installierte Applikationen
- Lokale Datenhaltung
- Individuelle Update-Zyklen
- Unterschiedliche Hardware-Generationen

Das Problem bei Scale

Viele Endgeräte multiplizieren den Aufwand erheblich:

- Viele Betriebssysteme gleichzeitig pflegen
- Viele Update-Rollouts koordinieren
- Viele lokale Datensicherungen absichern
- Hoher Support-Aufwand pro Device
- Sicherheitsrisiken

📄 Je mehr Endgeräte, desto größer der Overhead — VDI zentralisiert genau diesen Teil.

Der VDI-Gedanke

Anstatt jedes Endgerät individuell zu verwalten, wird der gesamte Workspace in die zentrale Infrastruktur verlagert. Das Endgerät wird damit zu einem reinen **Access Device** und es zeigt lediglich an, was zentral berechnet wird.



Zentrale Infrastruktur

Desktop, Anwendungen und Daten liegen zentral auf Servern — sicher, konsistent und zentral verwaltbar.



Browser-Zugriff

Der User benötigt nur einen Browser oder Thin Client — keine lokal installierte Software, kein aufwändiges Provisioning.



Thin Clients & Laptops

Unterschiedliche Endgeräte greifen auf dieselbe zentrale Umgebung zu — Hardware-Heterogenität wird irrelevant.

Aber VDI ist nicht gleich VDI

VDI ist kein monolithisches Konzept und je nach Anforderung kommen unterschiedliche Modelle zum Einsatz:

Persistent Desktop

Jeder User besitzt eine eigene dedizierte VM mit persistentem Zustand.
Änderungen bleiben erhalten.

User 1 → VM 101 · User 2 → VM 102

Non-Persistent Desktop

Desktops werden bei Login erzeugt und nach der Session verworfen.
Ressourceneffizienter, aber zustandslos.

Login → Erstellen → Session → Logout → Löschen

Application / Browser Workspace

Nicht der gesamte Desktop, sondern nur einzelne Applikationen oder Browser-Sessions werden containerisiert bereitgestellt.

User → Browser → Container/App

 Der Application/Browser-Workspace-Ansatz ist besonders ressourceneffizient — und genau hier wird es interessant.

VDI $\neq$ nur Windows Desktop

In vielen Köpfen ist VDI gleichbedeutend mit Windows, RDP und VMware Horizon. Diese Gleichsetzung greift zu kurz. Der eigentliche Begriff beschreibt die **Entkopplung des Workspace vom Endgerät** — und das ist technologieoffen.

Klassische Assoziation

- Windows Desktop
- RDP-Protokoll
- VMware Horizon / Citrix
- Persistente VMs

Moderner Workspace-Begriff

- Linux Desktop
- Browser-Session
- Development Environment
- Security Tool / Terminal
- Temporärer Arbeitsplatz
- Isolierte Applikation

Die eigentliche Kernfrage lautet daher: **Was braucht der User gerade als Workspace?**



VDI im Proxmox-Umfeld

Proxmox VE ist eine ausgereifte Open-Source-Virtualisierungsplattform mit einem umfangreichen Feature-Set. Sie bringt alle notwendigen Bausteine mit, um eine klassische VM-basierte VDI-Plattform zu betreiben.

Virtualisierung

- KVM/QEMU
- LXC Container
- Cloud-Init

Cluster & HA

- Multi-Node Cluster
- High Availability
- Live Migration

Storage & Netzwerk

- Shared Storage
- SDN
- Firewall

Automation

- REST API
- Backup
- Monitoring

Auf dieser Basis lässt sich ein vollständiger Proxmox-Cluster aufbauen, in dem VMs oder Container Agents auf mehrere Nodes verteilt werden. Solide, aber ohne nativen Workspace-Layer.

Die Herausforderung

Proxmox ist primär eine

Virtualization & Infrastructure Platform

Keine vollständige VDI-Plattform.

Was fehlt für vollständige VDI?

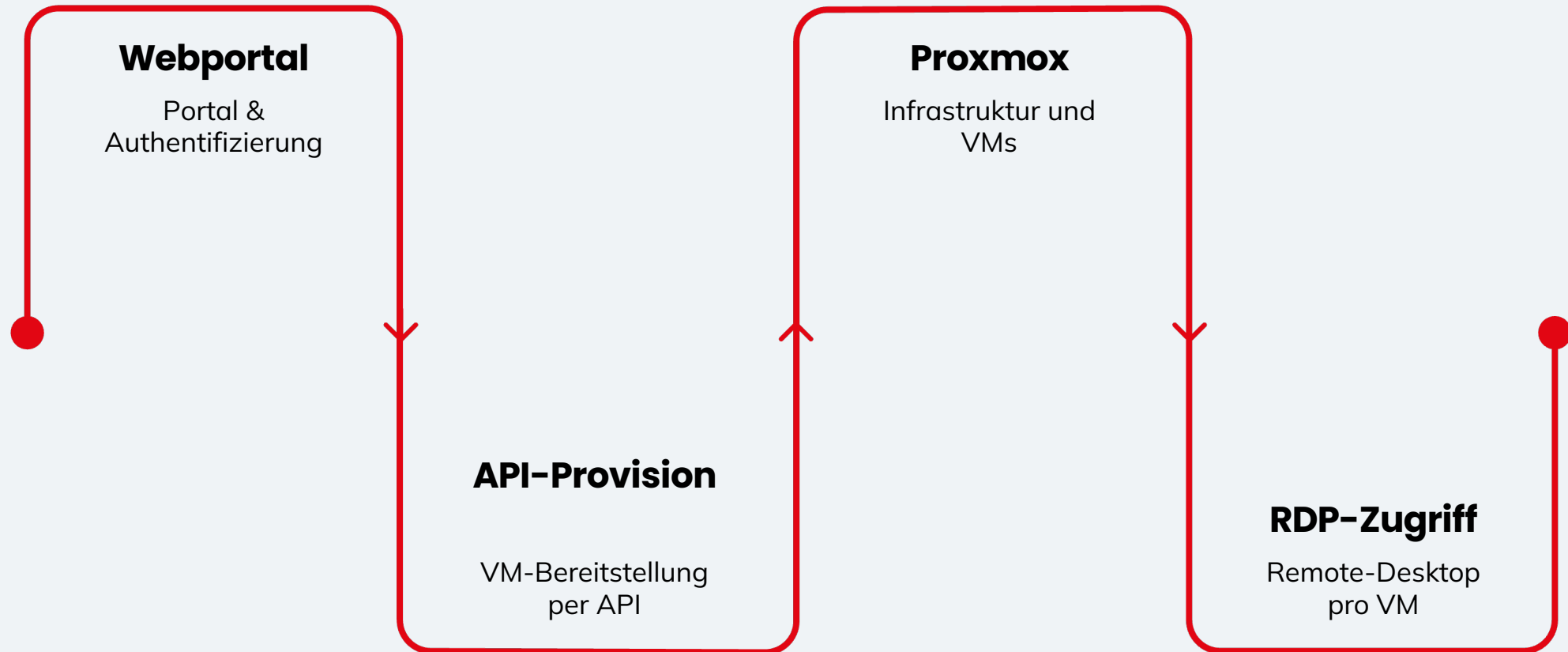
- User-facing Workspace-Portal
- Session Management
- Application Catalog
- Browser-basierter Zugriff
- Containerisierte Workspaces
- User Workspace Lifecycle
- Self-Service Experience

All das lässt sich selbst entwickeln...

Aber dann beginnt das eigentliche Projekt erst!

Der klassische Eigenentwicklungs-Ansatz

Wer VDI auf Basis von Proxmox vollständig selbst baut, muss einen erheblichen Entwicklungsaufwand einplanen. Ein typischer Stack könnte so aussehen:



Das funktioniert grundsätzlich — wirft aber eine entscheidende Frage auf: **Wie viele Komponenten müssen selbst entwickelt, gewartet und betrieben werden?** Portal, Auth, Provisioning-Logik, Session-Handling und Lifecycle-Management summieren sich schnell zu einem eigenständigen Produktprojekt.

Und dann kommt die Skalierung

Statische VDI-Kapazität ist kostspielig. Reale Nutzungsprofile sind selten gleichmäßig:

Reales Nutzungsprofil bei 500 Usern

- Nachts: ~5 aktive Sessions
- Tagsüber: ~50 aktive Sessions
- Morgens Spitze: plötzlich 150
- Einzelne Sessions ressourcenintensiver

500 VMs rund um die Uhr zu betreiben wäre ineffizient und teuer.

Der gewünschte Mechanismus

- 1 Bedarf erkennen**
- 2 Workspace erstellen**
- 3 User arbeitet**
- 4 Workspace entfernen**

 Das führt zum entscheidenden Konzept: **Elastic VDI** — Infrastruktur, die mit dem Bedarf wächst und schrumpft.

Was wäre also eine ideale Lösung?

Die Antwort liegt in der klaren Trennung von Verantwortlichkeiten auf drei Ebenen:

Infrastructure Layer

Proxmox VE

- Compute & RAM
- Storage
- Networking / SDN
- HA & Live Migration
- REST API

Workspace Layer

Kasm Workspaces

- Browser-basierter Zugriff
- User Sessions
- Application Catalog
- Container-Workspaces
- Authentication

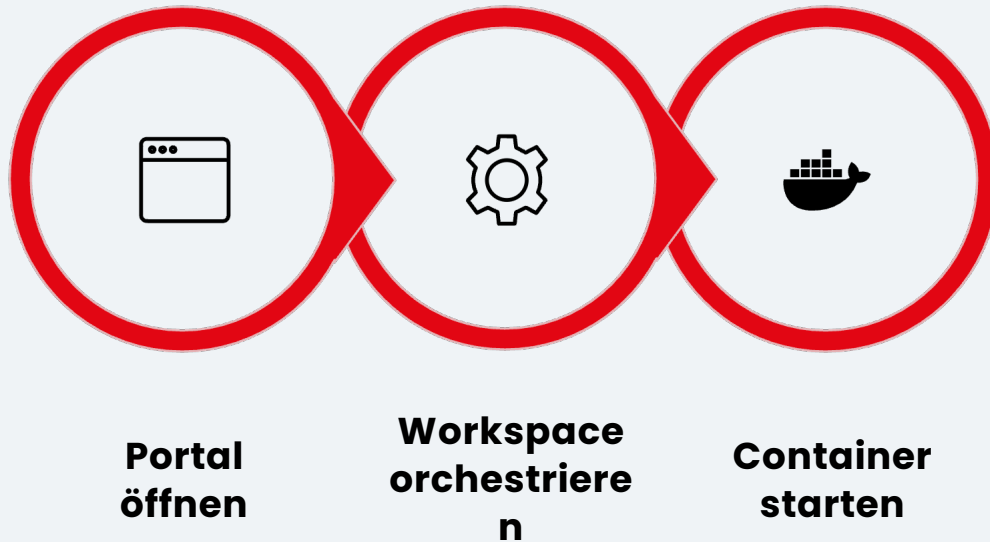
Automation Layer

Autoscaling

- Bedarf erkennen
- Workspaces starten
- Ressourcen hinzufügen
- Ressourcen entfernen

Kasm Workspaces

Kasm verfolgt einen grundlegend anderen Ansatz als klassische VDI. Anstatt jedem User eine vollständige VM zuzuweisen, wird der Workspace als isolierter Container bereitgestellt und ist direkt über den Browser zugänglich.



Keine klassische RDP-Client-Installation erforderlich. Der Zugriff erfolgt via HTTPS mit einem modernen Web-basierten Streaming-Protokoll. Das senkt die Einstiegshürde für Endgeräte und vereinfacht den Rollout erheblich.



Kasm Workspaces

Ein Workspace ist in Kasm nicht auf einen spezifischen Desktop-Typ beschränkt. Das System kann nahezu beliebige containerisierte Umgebungen bereitstellen:



Isolierte Browser

Firefox, Chrome — datenschutzkonform in isolierten Containern für sicheres Surfen ohne lokale Spuren.



Dev Environments

VS Code, Terminal — isolierte Entwicklungsumgebungen mit definierten Toolchains per Container Image.



Full Linux Desktop

Ubuntu, Kali Linux — vollständige Desktops für Entwicklung, Security-Analysen oder Training.



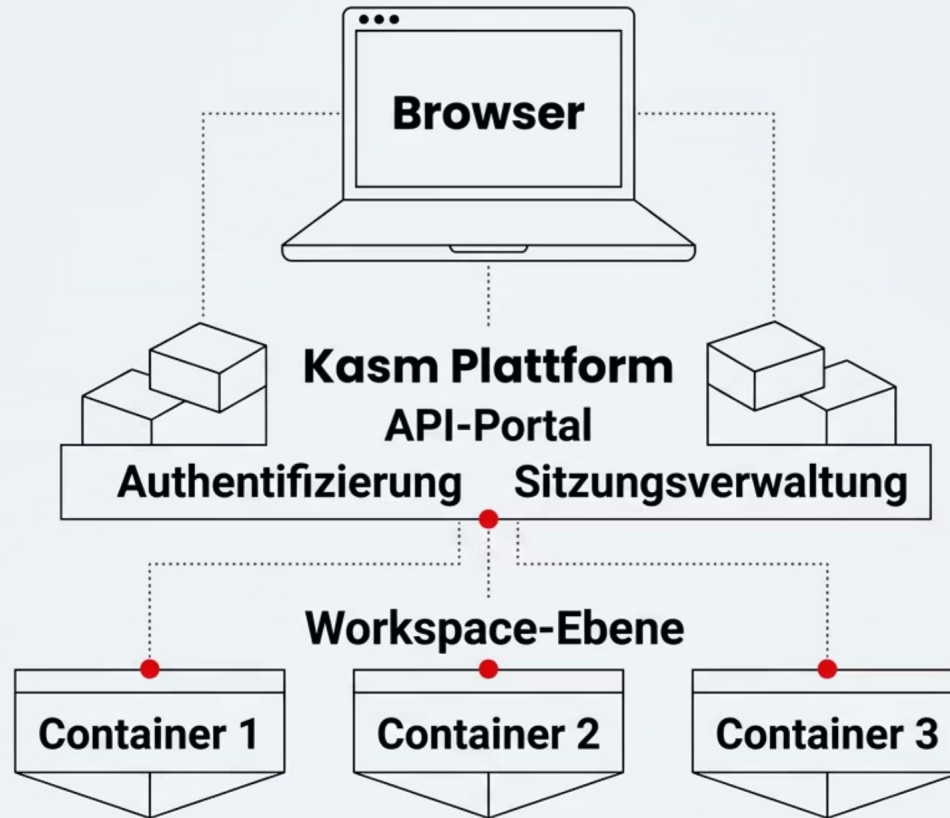
Productivity Apps

LibreOffice und weitere Office-Anwendungen als containerisierte, zentral verwaltete Workspaces.

Das Paradigma verschiebt sich: von „Ich brauche einen Desktop" zu „Ich brauche eine Umgebung für meine Aufgabe."

Die Kasm-Architektur

Kasm gliedert sich in einen zentralen API/Portal-Layer und dezentrale Workspace-Agenten, die die eigentlichen Container starten und verwalten.



Der Kasm-Core übernimmt Authentifizierung, Session-Routing und Lifecycle-Management. Die Workspace-Agenten können auf beliebigen Hosts betrieben werden — was die Integration mit Proxmox ermöglicht.

Kasm + Proxmox

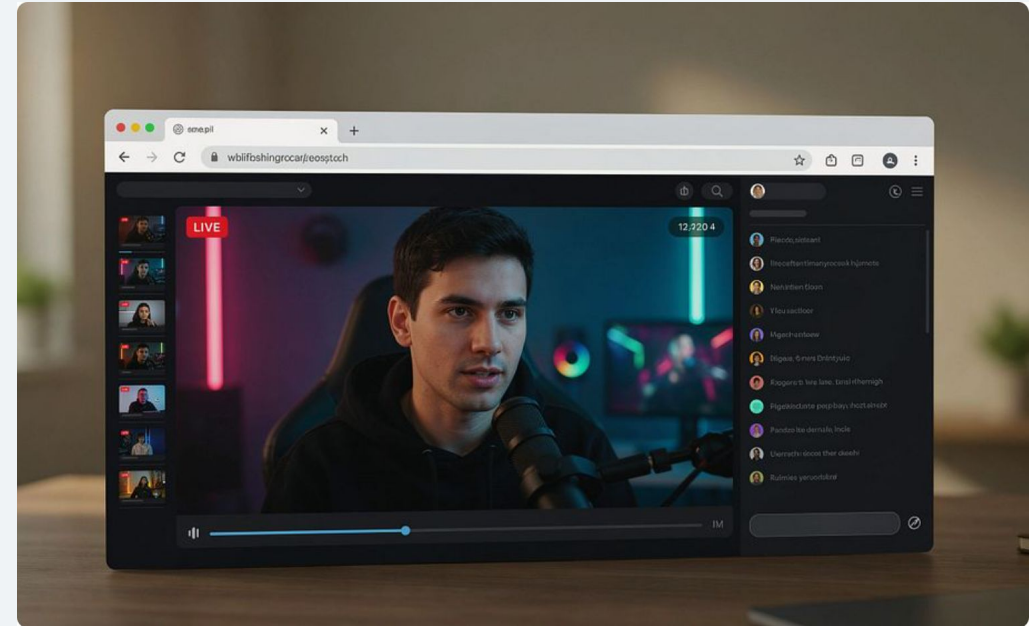
Die Kombination beider Systeme erzeugt eine leistungsfähige, vollständige VDI-Plattform: Proxmox stellt die Infrastruktur bereit, Kasm liefert den Workspace-Layer darüber.



Proxmox VMs als Agenten-Hosts

Auf Proxmox-VMs laufen Docker-Instanzen mit Kasm-Agenten. Proxmox verwaltet die VM-Infrastruktur vollständig über seine API.

Proxmox liefert die Infrastruktur. Kasm liefert den Workspace.



Kasm orchestriert Workspaces

Kasm platziert Container-Workspaces auf verfügbaren Agenten-VMs. Der User sieht nur den Browser-Zugriff — die Infrastruktur dahinter ist transparent.

Warum Container statt VM pro User?

VM-basierte VDI

-  **Vollständige VM pro User**
-  **Eigenes OS-Image pro Session**
-  **Langsamer Startvorgang**
-  **Hoher Ressourcen-Overhead**

Container-basierte Workspaces

-  **Schnellerer Startvorgang**
-  **Geringerer Overhead**
-  **Höhere Workspace-Dichte**
-  **Einfacher Lifecycle**

Container sind ideal für temporäre, zustandslose Sessions. Die Startzeiten liegen im Sekundenbereich — eine einzelne Proxmox-VM kann dabei viele simultane Workspace-Container hosten.

Der interessante Teil: Autoscaling

Elastic VDI

Was passiert, wenn plötzlich viele User gleichzeitig Workspaces anfordern? Statische Kapazität führt entweder zur Über- oder Unterauslastung. Autoscaling löst dieses Dilemma durch dynamische Bereitstellung.

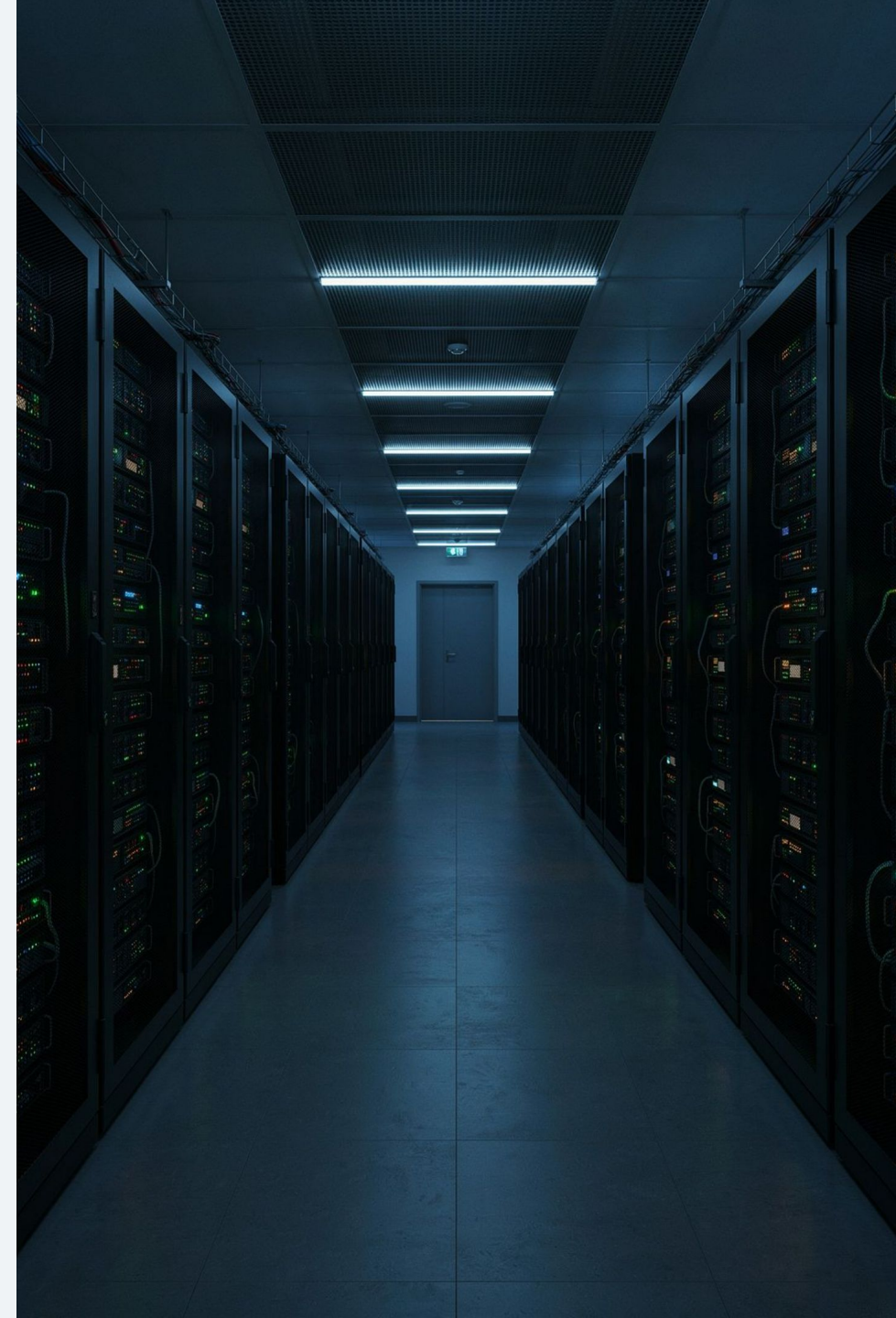
Nachfrage steigt

Autoscaler startet

Proxmox Provision

Kasm Agent startet

Die Infrastruktur wächst automatisch mit dem Bedarf — ohne manuellen Eingriff und ohne vorab provisionierte Überschusskapazität.



Scale Out

Sobald die Kapazitätsschwelle eines bestehenden Agenten überschritten wird, provisioniert der Autoscaler transparent eine neue VM auf dem Proxmox-Cluster und registriert sie als neuen Kasm-Agenten.

Ausgangszustand

3 Agents × 10 Workspaces = **30 Workspaces** gesamt

Nachfrage steigt

Auslastung nähert sich 100% der aktuellen Kapazität — Autoscaler wird ausgelöst.

Neuer Agent

Proxmox provisioniert neue VM → Kasm-Agent startet → 10 weitere Workspaces verfügbar.

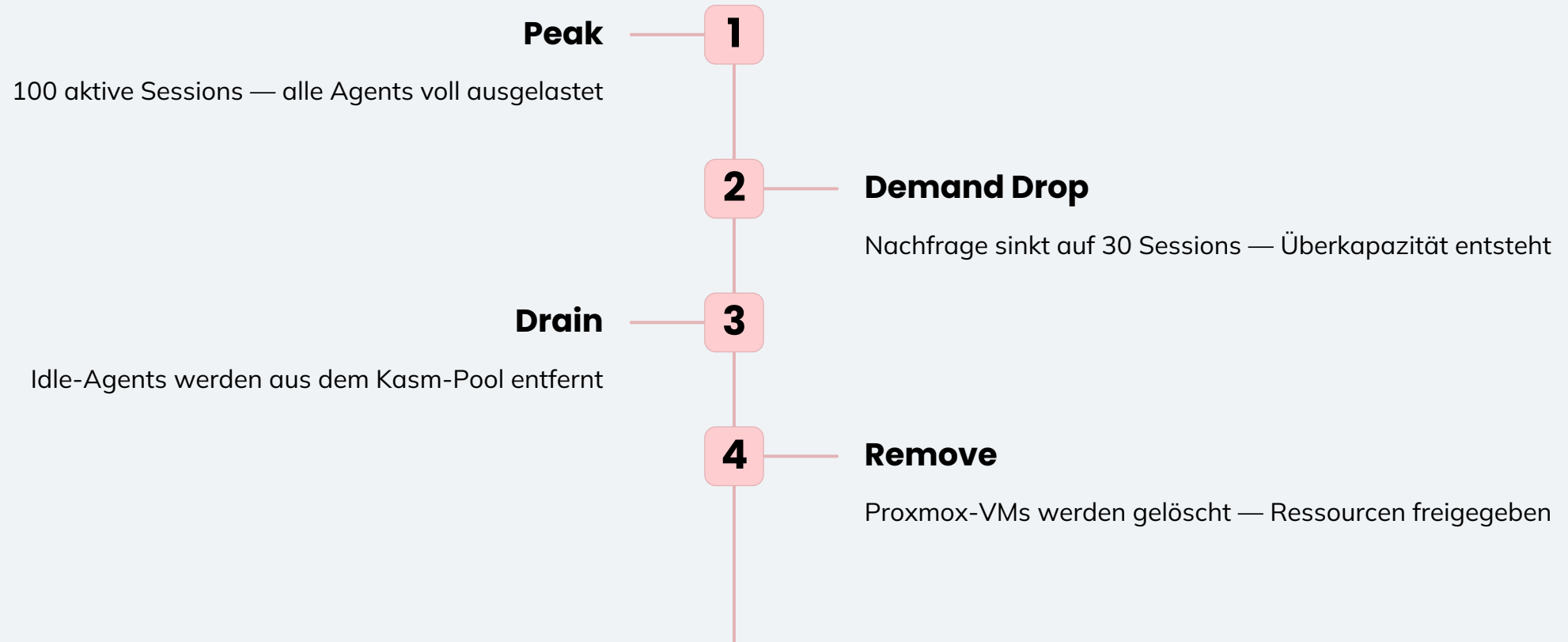
Erweiterter Zustand

4 Agents × 10 Workspaces = **40 Workspaces** gesamt

Neue Kapazität wird automatisch bereitgestellt und steht innerhalb weniger Minuten zur Verfügung.

Scale Down

Genauso wichtig wie das Skalieren nach oben ist das kontrollierte Zurückfahren nicht benötigter Kapazität. Ein intelligenter Autoscaler erkennt ungenutzte Agenten und entfernt sie geordnet.



Damit wird aus statischer VDI eine echte **Elastic VDI**: Die Infrastruktur kostet nur das, was gerade tatsächlich genutzt wird.

Das eigentliche Zusammenspiel

Die drei Ebenen sind klar voneinander getrennt — jede Komponente hat eine definierte Rolle:

Kasm entscheidet

Was bekommt der User?

User-Management, Sessions, Workspace-Typen, Authentication und Application Catalog liegen vollständig im Kasm-Layer.

Proxmox entscheidet

Wo läuft es?

VM-Placement, CPU/RAM-Scheduling, Storage-Zuweisung und Netzwerk-Routing — Proxmox verantwortet die physische Infrastruktur.

Autoscaling entscheidet

Wie viel Infrastruktur?

Der Autoscaler beobachtet Auslastungsmetriken und kommuniziert mit der Proxmox-API, um Agenten-VMs dynamisch zu erstellen oder zu entfernen.



Diese klare Separation of Concerns macht das System wartbar, erweiterbar und replizierbar.

Was bringt dieser Ansatz?



Für den User

- Browser genügt — keine lokale Installation
- Schneller, konsistenter Zugriff
- Zentrale, definierte Workspace-Umgebung



Für Operations

- Zentrale Verwaltung via API
- Proxmox HA, Backup & Monitoring
- Infrastructure as Code möglich



Für die Infrastruktur

- Ressourcen bedarfsgerecht bereitstellen
- Bessere Gesamtauslastung
- Scale-out und Scale-down automatisiert

Wo liegen die Grenzen?

Kasm + Proxmox ist ein leistungsfähiger Stack — aber kein universeller Ersatz für jede klassische VDI-Plattform. Folgende Aspekte sind vor dem Einsatz sorgfältig zu bewerten:

Technische Einschränkungen

- **GPU Workloads:** GPU-Passthrough möglich, aber komplex
- **Windows Desktops:** Kasm ist primär Linux-nativ
- **Multimedia:** Streaming-Protokoll hat Latenzen
- **Storage Performance:** I/O-intensive Workloads prüfen
- **Netzwerk-Latenz:** kritisch für interaktive Sessions

Organisatorische Aspekte

- **Identity Management:** LDAP/AD-Integration nötig
- **Persistence:** Zustandsbehaftete Workspaces erfordern Planung
- **Licensing:** Kasm hat Community & Enterprise Tiers

❏ Nicht jeder Workspace braucht dieselbe Architektur — Analyse der Use Cases ist entscheidend.

Wann ist dieser Ansatz interessant?



Temporary Workspaces

Kurzlebige, zustandslose Arbeitsumgebungen, die nach Nutzung automatisch verworfen werden. Kein Cleanup-Aufwand.



Training & Education

100 identische Kurs-Umgebungen auf Knopfdruck — nach dem Training vollständig entfernt. Ideal für Schulungszentren.



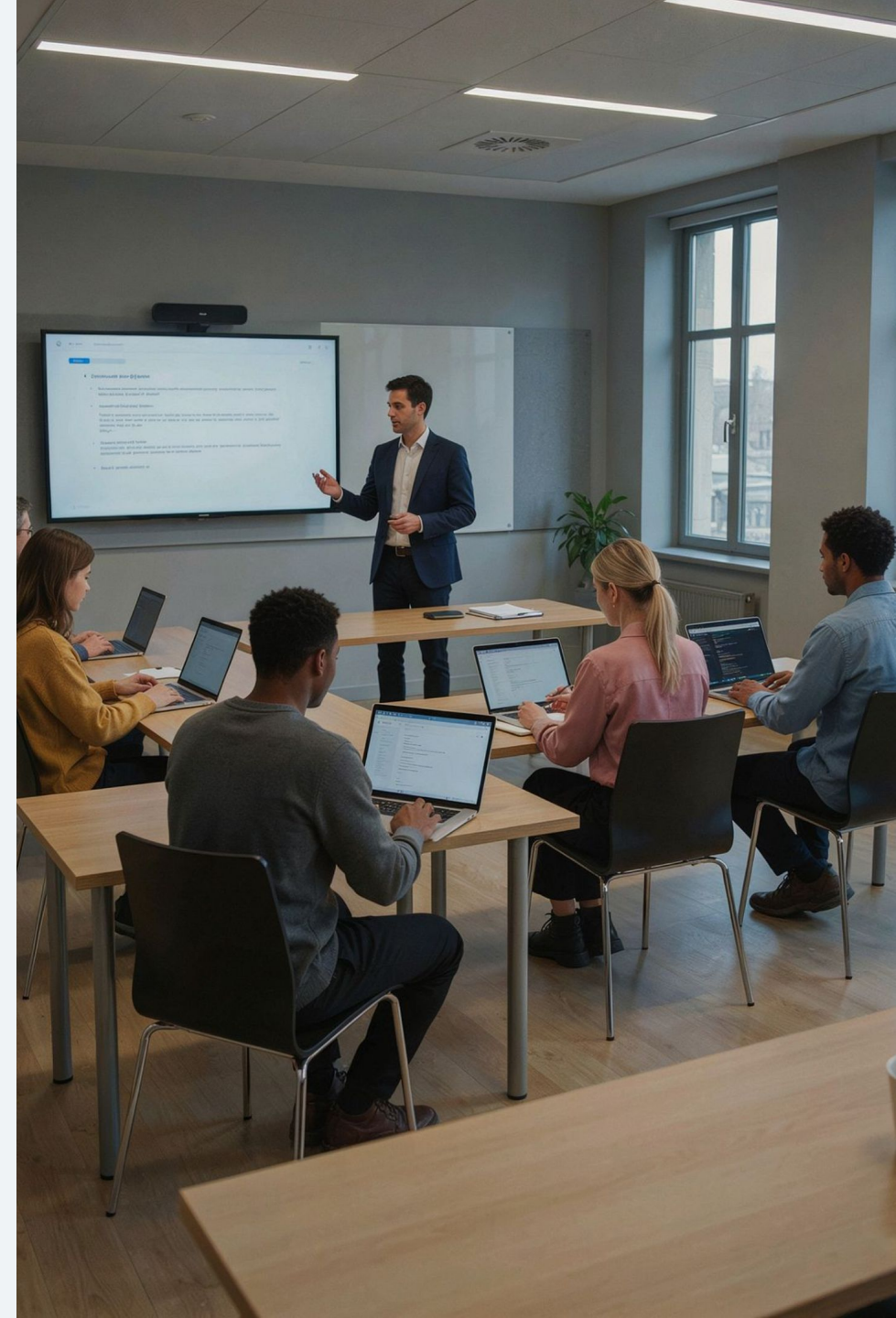
Development

Isolierte Dev-Environments mit definierten Toolchains — reproduzierbar, wegwerfbar, zentral bereitgestellt.



Secure Browsing

Isolierte Browser-Containers für sicheres Surfen im Internet — keine lokalen Artefakte, kein Risiko für das Endgerät.



Demo

1

Proxmox Node

Ein einzelner PVE-Node

Auto-Scale

Workspace Kapazität

Automatisch skalierende Agents
(Docker Agents in VMs)

∞

Sessions

Sessions skalieren anhand der auto-scale
Konfiguration.

KASM Web:	https://vdi01.lab.gyptazy.com	Proxmox VE Web:	https://pve03.lab.gyptazy.com:8006
KASM User:	admin@kasm.local	PVE User:	root@pam
KASM Password:	JOcoJGzagkPZ2	PVE Password:	demo01

Takeaways

VDI ist nicht nur eine VM

Ein moderner Workspace kann vieles sein — und sollte an den tatsächlichen Bedarf des Users angepasst werden, nicht umgekehrt:

VM

Persistenter, vollständiger Desktop

Container

Schnell, leichtgewichtig, zustandslos

Application

Einzelne App, kein vollständiger Desktop

Browser

Isolierte Web-Session, kein Client nötig

Dev Environment

Reproduzierbar, definiert, wegwerfbar

Proxmox liefert die Infrastruktur

Proxmox VE ist die solide, API-getriebene Grundlage, auf der die gesamte Autoscaling-Logik aufsetzt. Als vollständige Open-Source-Plattform ohne Vendor Lock-in ist sie ideal für diesen Einsatzzweck.

Compute

KVM/QEMU und LXC für flexible VM- und Container-Workloads auf x86-Hardware

Storage

Unterstützung für lokale, NFS-, Ceph- und iSCSI-Backends — flexibel kombinierbar

Network & SDN

Software Defined Networking, Firewall und VLAN-Support für saubere Netzwerk-Segmentierung

HA & API

High Availability, Live Migration und eine vollständige REST API für Automatisierung und IaC

Kasm liefert den Workspace

Kasm Workspaces transformiert containerisierte Umgebungen in nutzerfreundliche, browser-zugängliche Workspaces — mit vollem Session-Management und User-Portal out of the box.

Sessions & Auth

Integriertes Session-Management mit LDAP/SAML/OIDC —
Single Sign-On ready

Browser Access

Streaming-Protokoll für niedrige Latenz — nur Browser
erforderlich, kein Client-Install

Application Catalog

Flexibler Workspace-Katalog mit vordefinierten und custom
Container-Images

User Experience

Self-Service-Portal für End-User — konsistente UX unabhängig
vom Endgerät

Alternative: Bekane (Open-Source)



Open-Source

Vollständig Open-Source
Enterprise Support durch FactorFX

Native Proxmox VE Integration

Nahtlose Integration von VMs, Templates und Snapshots für
Proxmox VE Umgebungen.

Application Catalog

Flexibler Workspace-Katalog mit vordefinierten und custom
Container-Images

Anwendungszwecke

Fokus auf Umgebungen zum Lernen, Remote & Hybrid Work,
Softwareentwicklung, als auch Training- & Workshops.

Informationen

Ankündigung: <https://www.factorfx.com/en/bekane-open-source-vdi-solution/>
Docs: <https://documentation.bekane.org/project-overview/introduction>
Source: <https://gitlab.com/factorfx/bekane>

Danke! Fragerunde!

Vielen Dank für Ihre Aufmerksamkeit. Gerne beantworte ich jetzt Ihre Fragen!



Florian Paul Azim Hoberg

@gyptazy



Email

florian.hoberg@credativ.de



GPG

0xB1B88CBB