UNIVERSITY OF ŽILINA
Faculty of Management Science and Informatics

BRAIN:IT

PROJECT

Multi-tenant cybersecurity operations centre designed
as an open cloud service with machine learning elements

V1 – Result of the analysis of available solutions

KPB1 – Design of the virt. solution

type – documentation

delivery month – M3

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

Contents

1 Introduction ... 3
2 Selection of candidate solutions ... 3
2.1 Overview of suitable types of solutions ... 3
2.1.1 Open Source Cloud Management Platforms (CMP) – Full-fledged IaaS solutions ... 3
2.1.2 Hypervisor-centric Virtualization Platforms with Cloud Management elements 4
2.1.3 Container Orchestration Platforms (with potential for VMs) ... 4
2.2 Conditions for the selection of candidates ... 5
2.2.1 Narrowing down the selection ... 5
3 Analysis of candidates for the cloud/ virtualization platform ... 6
3.1 Functional criteria ... 6
3.1.1 K1: Support for IaaS functionality (weight: 5) ... 6
3.1.2 K2: Modularity and architecture (weight: 4) ... 9
3.1.3 K3: Support for orchestration and automation (weight: 4) ... 13
3.2 Technical criteria ... 15
3.2.1 K4: Performance and scalability (weight: 5) ... 15
3.2.2 K5: Reliability and stability (weight: 4) ... 21
3.2.3 K6: Compliance with standards and interoperability (weight: 4) ... 24
3.3 Operational criteria ... 27
3.3.1 K7: Complexity of deployment and administration (weight: 3) ... 27
3.3.2 K8: Support, availability of documentation and community (weight: 4) ... 30
3.4 Security criteria ... 33
3.4.1 K9: Authentication and access control (weight: 5) ... 33
3.4.2 K10: Tenant isolation and data protection (weight: 5) ... 36
3.4.3 K11: Network/monitoring integration (weight: 3) ... 40
4 Evaluation table for the selection of the cloud platform ... 43
5 Final summary ... 45
List of sources ... 46

1 INTRODUCTION

In the current environment of constant cyber threats, the **Multi-tenant cybersecurity operations centre (SOC) designed as an open cloud service with machine learning elements** represents an infrastructure for improving the protection of digital assets. The implementation of such a centre requires an appropriate selection of an adequate technological platform.

This document provides a comprehensive analysis and comparison of selected open-source solutions that are suitable as virtualization and cloud platforms for building the intended solution. The main objective is the deployment of a multi-tenant environment, ideal for the provision of SOCaaS (Security Operations Center as a Service) services. The analysis is based on the specified functional, technical, operational and security criteria that were defined in the previous output (V2 – Definition of selection criteria).

The purpose of this analysis is to identify and evaluate the strengths and weaknesses of the analysed solutions. Emphasis is placed on their suitability for deployment in an academic environment, which typically imposes specific requirements on openness, flexible scalability, a high level of security, simple administration and cost-effectiveness, but which also takes into account the needs of public administration. This analysis is to provide a solid basis for an informed decision when selecting a suitable platform for the implementation of the project.

2 SELECTION OF CANDIDATE SOLUTIONS

The aim of identifying candidate solutions is to select a platform capable of providing the intended IaaS (Infrastructure as a Service) services. The deployment of the INNOSOC service requires properties such as high availability, the ability to isolate tenants and comprehensive security mechanisms. We give preference to open-source solutions that are suitable for the academic and research sphere.

2.1 OVERVIEW OF SUITABLE TYPES OF SOLUTIONS

On the basis of these requirements and the preceding discussion, we have identified three key categories of solutions, from which we will select a shortlist of candidates for detailed analysis.

2.1.1 Open Source Cloud Management Platforms (CMP) – Full-fledged IaaS solutions

These platforms are designed specifically for building and managing private clouds. They offer a complete set of services for the virtualization of compute resources, network functions and storage, including APIs for automation and orchestration. They focus on multi-tenant environments and provide a high degree of isolation. These solutions include:

- **OpenStack** – an open-source cloud system that provides extensive capabilities for scaling, tenant isolation, network virtualization and modularity. For us, it is the baseline reference solution.
- **OpenNebula** – An alternative with a flexible architecture and lower complexity, suitable for smaller and academic environments.
- **Apache CloudStack** – Robust and easily scalable.

- **VirtEngine** – A less well-known open-source platform that combines management of VMs, containers and billing; however, it has a limited community and weaker support.
- **Eucalyptus** – an open-source alternative to Amazon EC2

Commercial solutions in this category include:

- **Microsoft Azure Stack HCI** – A hybrid solution enabling on-premise environments to be extended into the public Azure cloud. Suitable mainly for Microsoft-centric environments.

2.1.2 Hypervisor-centric Virtualization Platforms with Cloud Management elements:

These are solutions that primarily serve to manage hypervisor-based virtual machines. They offer web interfaces, clustering functions and APIs that enable efficient management and scaling of the virtual infrastructure. Although they may not have a full-fledged IaaS-level "cloud" abstraction like CMPs, they are extremely powerful and often simpler to deploy and administer. These solutions primarily provide VM virtualization but can be extended with cloud functions.

- **Proxmox VE** – An easily deployable, open-source platform supporting both VMs and containers. Suitable for rapid deployments. It has an active community and suitable tools for HA and clustering.
- **oVirt** – An enterprise virtualization platform from Red Hat built on KVM. It provides tools for virtualization management, but less developed cloud functions and multitenancy.
- **XCP-ng** - is an open-source virtualization platform built on the Xen hypervisor, intended for deploying VMs on physical hardware. In combination with **Xen Orchestra**, it offers a complete web interface for the management, backup and monitoring of the infrastructure. This pair forms a robust, scalable and licence-free solution suitable for both companies and the academic environment.

Commercial solutions in this category include:

- **VMware vSphere** – One of the most widespread virtualization systems, with comprehensive support for HA, DR, networking and management. High licensing demands.
- **Nutanix AHV** – A hyperconverged solution that combines compute, storage and management. Strong integration, but a closed ecosystem.

2.1.3 Container Orchestration Platforms (with potential for VMs):

These platforms are dominant in the field of containerization and microservices. Although they are not primarily IaaS with VM support, their orchestration, scaling and container-level isolation capabilities are relevant to modern architectures. Some of them offer options for running virtual machines within a container environment. Their inclusion is important in order to take future trends and architectures into account. In this category, Kubernetes solutions are extremely widespread. Their deployment is more complex and requires a different architecture.

- **Kubernetes + KubeVirt** – The standard for container management, with the option of extending it with VM support using KubeVirt. It may be of interest for containerized SOCaaS, but it requires an entirely different architecture from a classic IaaS environment.
- **Docker Swarm** – a native orchestration solution for Docker containers that makes it easy to create and manage clusters. It enables service scaling, automatic load

balancing and high availability. Thanks to its simple CLI and integration with the Docker ecosystem, it is suitable for small to medium-sized deployments. It is an alternative to Kubernetes for users who prefer simplicity.

- **Nomad** – is a lightweight and flexible orchestrator from HashiCorp, intended for deploying and managing both containerized and non-containerized workloads (e.g. binaries, VMs, JAR files). It is characterized by a simple single-binary architecture and high scalability. It supports integration with Consul and Vault for service discovery and security. It is suitable for teams looking for a less complex alternative to Kubernetes.
- **Rancher** – an open-source platform for managing Kubernetes clusters that simplifies their deployment, monitoring and management from a single central interface. It supports multiple clusters at once, including those running in the cloud, on-premise or in a hybrid setup. It provides tools for access management, CI/CD, monitoring and security. It is ideal for teams that want centralized Kubernetes management without complex configuration.

Commercial solutions in this category include:

- **Red Hat OpenShift Virtualization** – Built on Kubernetes with added VM support. Strong for containerized applications, less so for traditional IaaS.

2.2 CONDITIONS FOR THE SELECTION OF CANDIDATES

When selecting candidates for comparison in the next phase of the project, the following basic selection conditions were set:

- The platform must be **open-source**, available under a free licence and without any obligation to commit to a specific commercial vendor,
- It must be **deployable on-premise**, with the possibility of operation in the environment of an academic institution or an internal data centre,
- support for IaaS functionality,
- the ability to create a multi-tenant environment,
- simplicity or feasibility of deployment and administration,
- suitability for use in the environment of a SOCaaS-type security service.

The aim is to find a solution that provides IaaS (Infrastructure as a Service) functionality, enables a multi-tenant environment and is sufficiently robust for security operations.

2.2.1 Narrowing down the selection

On the basis of these aspects, the following open-source platforms were selected as the **most suitable candidates for detailed analysis**:

- **OpenStack** [1]**: A** comprehensive IaaS platform with a modular architecture and extensive support for virtualization, network segmentation, tenant isolation and automation. It is the most widespread open-source cloud, with an active community. Its advantages are high flexibility and integration with standards. A disadvantage may be the complexity of deployment.
- **Apache CloudStack** [2]– A robust but less widespread solution, with a lively but smaller community and slower development dynamics. For this project, from the perspective of deployment flexibility and the ecosystem on offer, it was ultimately

preferred over OpenNebula. In the case of simpler academic projects, this would be reversed in favour of OpenNebula [1] [2].

- **Proxmox VE** [5]**:** A virtualization platform with an emphasis on simplicity and efficiency. It is extremely popular in academic and testing environments. It supports VMs and containers and has a web interface and support for clustering. Although it is not a full-fledged IaaS solution, its architecture enables some cloud functionalities with less technical debt. However, it is limited in terms of native multitenancy and API orchestration.
- **Kubernetes** [6]**:** Although it does not support IaaS and VMs, it has been included in the selection as a currently important solution for running modern applications. If there were a requirement for "containerized SOCaaS", it would be a clear candidate, but for "cloud or virtualization solutions" in the context of VMs it is less suitable.

Candidates not selected and reasons for exclusion:

- **OpenNebula** [7]**:** A simpler and more accessible alternative to OpenStack. It enables flexible deployment in smaller and academic environments, supports various hypervisors and offers a reasonable degree of multitenancy, albeit at a lower level. It is less demanding to install and administer, but provides fewer advanced functionalities than OpenStack or CloudStack. It has a smaller community [1] [2]. However, it is less suitable for deploying and operating state administration services.
- **oVirt** [8]– Focused primarily on internal virtualization; multitenancy and IaaS functionality are limited.
- **Eucalyptus** [9]: The project has been inactive for a long time, development is stagnating, the website is minimally maintained and the community is practically non-existent. From the sustainability perspective: the platform has no prospect of further development, and its deployment would be risky in terms of security, support and compatibility with modern standards.
- **Harvester** [10]**, KubeVirt** [11]**, Docker Swarm, Nomad, Rancher** - container-oriented solutions. Their deployment would require a fundamental change to the project architecture, and they do not directly meet the IaaS requirements.
- **Commercial solutions (VMware, Nutanix, Azure Stack, OpenShift Virtualization)** – Excluded because of licensing policy, costs or dependence on closed ecosystems, which are not suitable for an open-source-oriented research project.

The next section presents a detailed analysis of the three selected solutions (OpenStack, CloudStack, Proxmox VE, Kubernetes) according to a set of established evaluation criteria.

3 ANALYSIS OF CANDIDATES FOR THE CLOUD/ VIRTUALIZATION PLATFORM

3.1 FUNCTIONAL CRITERIA

3.1.1 K1: Support for IaaS functionality (weight: 5)

Rationale: The project requires full IaaS functionality – management of virtual machines (VMs), disks, networks, images, snapshots and separate resources. The platform must support the creation and management of various instance types, allow resources to be allocated according to tenants' needs and, at the same time, offer a robust API for

programmatic access. The ability to provide complete IaaS functionality is key to ensuring the flexible and reliable operation of SOC as a service.

Scoring recommendation:

- 5 points: Support for complete IaaS functions (VMs, networks, disks, snapshots), strong APIs, confirmed references.
- 3 points: Most of the basic IaaS functions available, API with certain limitations.
- 1 point: Limited IaaS functionality, weak or no API support.

3.1.1.1 OpenStack

Description of functionality: OpenStack is currently the de facto standard for open-source IaaS cloud. Its architecture is modular and comprises specialized components for various IaaS functionalities. OpenStack is deployed globally in hundreds of production environments, including large enterprises, telecommunications operators and research institutions. Its IaaS functionality has been proven in large-scale and critical deployments.

- **VMs (Compute – Nova):** Full VM life cycle management (starting, stopping, pausing, scaling, migration), support for various hypervisors (KVM, vSphere, Hyper-V, Xen).
- **Disks (Block Storage – Cinder):** Dynamic creation and attachment of block storage to VMs, support for various backends (LVM, Ceph, NetApp, EMC, etc.), the option of creating disk snapshots.
- **Networks (Networking – Neutron):** Advanced network virtualization, including the creation of virtual networks, subnets, routers, security groups (VM-level firewall), floating IP addresses and VPN. Support for various network plugins.
- **Images (Image Service – Glance):** A central repository for VM images (VM templates), enabling their management and versioning.
- **Snapshots (Nova, Cinder):** Support for snapshots of VMs (via Nova) and disks (via Cinder) for backup and recovery purposes.
- **API:** Each OpenStack component provides a robust and well-documented RESTful API that enables full programmability and integration with external tools. There is also extensive support for CLI and SDKs.

Rationale: OpenStack is a comprehensive IaaS solution with full support for the management of VMs (Nova), disks and storage (Cinder, Swift), networking (Neutron), image services (Glance), snapshots and identity management (Keystone), as well as additional components for orchestration and fault management, which ensures high availability [12]. Thanks to its extensive modularity, OpenStack can control large compute, storage and network resources, which can be dynamically scaled and automated through the dashboard or the standardized REST API, as also confirmed by local deployments in the academic sphere ([12] [13]). OpenStack is considered the de facto standard for open-source clouds.

Score: 5 pts

3.1.1.2 CloudStack

Description of IaaS functionality: CloudStack is an open-source IaaS platform that focuses on ease of use and scalability [2]. CloudStack is also used in production environments; its ecosystem and community are smaller than those of OpenStack. It is popular for its reliability and simplicity.

- **VMs:** It offers comprehensive VM life cycle management and supports various hypervisors (KVM, VMware ESXi, XenServer, Hyper-V). It enables the definition and management of compute and storage "offerings" (service offerings).
- **Disks:** It provides management of data storage for VMs, including persistent disks, disk snapshots and VM templates.

- **Networks:** It offers network virtualization with support for isolated virtual networks (VLAN, VxLAN), firewall rules (security groups), IP addresses and load balancing. The network functions are integrated and simpler to configure than in Neutron (OpenStack).
- **Images:** It supports the management of VM templates (images) and ISO files.
- **Snapshots:** It provides the option of creating snapshots of VMs and disks.
- **API:** CloudStack has a robust RESTful API that enables full control over IaaS resources and the automation of operations.

Assessment: CloudStack offers integrated management of VMs, networking, storage, images and snapshots. It provides a simpler interface (both UI and API) than OpenStack, while its functions are still suitable for enterprise deployments as well. A limitation may be its lower degree of modularity, but it does not lag significantly behind in the basic IaaS functions. It has a smaller community, a smaller ecosystem of add-ons and less flexibility for future extensions, which is due to its architecture.

Score: 4.5 pts

3.1.1.3 Proxmox VE

Description of functionality: Proxmox VE is an open-source virtualization platform that focuses primarily on the management of virtual machines and containers, with integrated functions for clustering and storage. It is not a full-fledged cloud platform in the sense of a comprehensive IaaS framework with native support for multitenancy, IaaS catalogues, advanced cloud services and virtualization like OpenStack. Very popular in SMB, academic and research environments for its simplicity, performance and openness:

- **VMs (KVM):** Full management of VMs (creation, starting, migration, backup, restoration) and LXC. It supports various operating systems and has good resource management.
- **Disks:** Full storage management for VMs and LXC containers, including local (LVM, ZFS) and network storage (NFS, SMB/CIFS, iSCSI, Ceph). It supports the creation and attachment of virtual disks.
- **Networks:** Basic network configuration for VMs (bridges, VLAN tagging). It lacks advanced network virtualization features such as virtual routers, platform-level managed security groups or automated IP address allocation for tenants. These would require manual configuration or external tools.
- **Images:** It supports the management of ISO images for OS installation and of VM/LXC container templates.
- **Snapshots:** Full support for storage-level snapshots of VMs and containers.
- **API:** Proxmox VE has a well-documented RESTful API that enables the automation of many VM, storage and cluster management tasks, as well as web-based control. The API is less standardized and offers fewer integration options.

Assessment: Proxmox VE provides functions for the management of VMs, disks, images and snapshots. It has a robust API. However, its "IaaS" functionality is limited in the area of advanced network virtualization and native cloud-level multitenancy support (absence of, e.g., projects, quotas and comprehensive network services for tenants such as those in OpenStack/CloudStack). These missing "cloud-native" IaaS elements would have to be built on top of Proxmox manually or using external tools, which reduces its ability to provide a *complete* IaaS service out of the box for a multi-tenant environment.

Score: 3 pts

3.1.1.4 Kubernetes

Description of IaaS functionality: Kubernetes is primarily a platform for container orchestration (PaaS – Platform as a Service) and microservice management. Its main goal is the automation of the deployment, scaling and management of containerized applications. It is not designed as an IaaS platform for virtual machines and their direct management. The dominant platform for containerized applications:

- **VMs:** Kubernetes does not natively manage virtual machines. However, there are projects such as **KubeVirt** that make it possible to run and manage VMs as containers within a Kubernetes cluster. This, however, is not native IaaS functionality of Kubernetes itself.
- **Disks:** Kubernetes has robust persistent storage management for containers (Persistent Volumes, Persistent Volume Claims), but this is intended for containers, not for traditional block disks directly for VMs (without KubeVirt).
- **Networks:** Kubernetes provides an advanced network abstraction for containers (Service, Ingress, Network Policies), but this is different from network virtualization for VMs in IaaS platforms. For VMs, it would be necessary to rely on the underlying IaaS or on a complex KubeVirt solution.
- **Images:** It manages container images, not VM images.
- **Snapshots:** It supports snapshots of persistent container volumes, not of VMs.
- **API:** Kubernetes has a powerful and flexible API for container orchestration. However, for IaaS functions for VMs, the KubeVirt API, which is an add-on layer, would have to be used.

Rationale: Kubernetes is not an IaaS platform. Its ability to manage VMs depends on add-ons such as KubeVirt, which is not native functionality and introduces considerable complexity. It does not meet the requirement for "complete IaaS functions" for VMs out of the box. Although it is key for modern applications, it is unsuitable for directly hosting SOCaaS at the IaaS level (as VMs) without substantial additional modifications and integrations. Kubernetes is regarded as PaaS or CaaS (Container as a Service) rather than IaaS, and it requires an IaaS layer beneath it to provide resources.

Score: 1 pt

3.1.1.5 Evaluation

Criterion K1 requires full IaaS functionality, including the management of virtual machines (VMs), disks, networks, images, snapshots and separate resources, with a robust API and confirmed references. The scores are as follows:

- **OpenStack**: 5 points – It provides complete IaaS functions, including VMs, networks, disks and snapshots, with a strong API and wide adoption.
- **CloudStack**: 4.5 points – Similarly to OpenStack, it offers full IaaS functionality with an API and references.
- **Proxmox**: 3 points – It supports VMs, disks, networks and snapshots and has an API, with confirmed references as an IaaS platform.
- **Kubernetes**: 1 point – It is not an IaaS platform; it serves to manage containers, not to provide infrastructure.

3.1.2 K2: Modularity and architecture (weight: 4)

Rationale: The modularity of the architecture fundamentally affects the extensibility and scalability of the solution. A platform that is divided into separate components (e.g. Nova – compute, Neutron – networking, Cinder – block storage) enables independent deployment,

tuning and replacement of modules as needed. It is also advantageous if the platform supports a modern microservice architecture, plugins, containerization and orchestration layers. Higher modularity also reduces the risk of an outage of the entire system when a single service fails.

Scoring recommendation:

- **5 points:** High modularity, microservices, separately deployable components, clear plugin integration.
- **3 points:** Modularity is partial; some components are mutually dependent.
- **1 point:** Monolithic architecture, limited options for modifying or replacing components.

3.1.2.1 OpenStack

Description: OpenStack is designed as a set of mutually interacting but separate projects (components), each of which provides a specific service. Each main component (e.g. Nova for compute, Neutron for networking, Cinder for block storage, Swift for object storage, Glance for images, Keystone for identity) has its own API and can be deployed, scaled and maintained independently of the others. Each component runs as a separate daemon or service, and they communicate with one another via defined APIs and message queues, thereby forming a unified cloud system. This microservice philosophy makes it possible to deploy only the components that are needed and to replace or scale them as required. OpenStack also supports a plugin architecture within individual services – for example, Neutron allows various network drivers (ML2 plugins), Cinder has drivers for various storage platforms, etc. Thanks to its open code, the system is highly extensible; the community adds new projects (e.g. Magnum for containers, Ironic for bare metal, Heat for orchestration) without the need to change the core architecture. The architecture therefore provides extreme flexibility and customizability – an organization can select and configure only a subset of services according to its needs. This decentralized architecture significantly increases the resilience and flexibility of the system [15] [16]. OpenStack continuously adopts modern approaches, including the containerization of its control services (e.g. via Kolla-Ansible or OpenStack-Helm), thereby further increasing modularity and simplifying management [17]. OpenStack supports:

- **Modularity**: It is divided into separate services, such as Nova (compute), Neutron (networking) and Cinder (storage), which can be deployed independently. This design enables flexibility and scalability, which is key for the INNOSOC project.
- **Microservices**: OpenStack uses a microservice architecture in which each service is a separate application with its own life cycle, which reduces the risk of an outage of the entire system.
- **Separately deployable components**: Yes, each service can be deployed and managed separately, which enables independent tuning and replacement of modules.
- **Plugin integration**: It supports a wide range of plugins and integrations, for example for hypervisors, storage and networking, which increases customizability.
- **Containerization and orchestration**: Through the Magnum component, it supports containers and orchestration with Kubernetes, which is in line with the project requirements.

Rationale: OpenStack is designed as a fully modular system. Its components are separate microservices with their own APIs, which enables flexible deployment, scaling and resilience to failure. Plugin support (especially in Neutron) and the trend towards containerization of control services reflect the needs of modern solutions. **The disadvantage** is high complexity: modularity brings with it more demanding deployment and administration. Overall, however, the architecture provides maximum flexibility and customizability.

Score: 4.5 pts

3.1.2.2 CloudStack

Description: CloudStack also has a modular architecture, although it is often described in the literature as "simpler" or "more monolithic" compared with OpenStack [18]. The main components are the CloudStack Management Server, which manages the entire cloud, and the CloudStack Agents, which run on the hypervisors. Nevertheless, services such as compute, networking and storage are logically separated and managed by the management server. CloudStack supports plugins for hypervisors (VMware, KVM, XenServer), storage and network services, which enables a certain degree of customization and extensibility [18]. However, it is not primarily built on a microservice architecture to the same extent as OpenStack, and the system components are not as strictly separated as in OpenStack. Comparative studies of OpenStack and CloudStack often point out that while CloudStack is simpler to deploy, its modularity is less granular than that of OpenStack. Thanks to its integrated architecture, CloudStack is installed as a single product, which considerably simplifies deployment and upgrades – the system is updated centrally and all parts are designed to be mutually compatible. CloudStack provides:

- **Modularity**: It has a modular architecture with components such as the Management Server, the Agent and plugins for hypervisors and storage. However, it is not as strictly microservice-based as OpenStack, which may limit its flexibility.
- **Microservices**: It supports some aspects of microservices, but not as extensively, with most functions integrated into the Management Server.
- **Separately deployable components**: Most components can be deployed separately, but there are certain dependencies, for example between the Management Server and the Agents, which may increase the risk of an outage.
- **Plugin integration**: It supports plugins for hypervisors (e.g. VMware, KVM) and storage, but it is less flexible than OpenStack.
- **Containerization and orchestration**: It has limited native support for containers, although it can be integrated with external orchestrators such as Kubernetes.

Rationale: CloudStack offers a simple, integrated architecture with multi-tenancy support, which is a plus for reliability and easy administration in a security-oriented cloud. Owing to its lower complexity, there is a lower likelihood of configuration errors or problems during upgrades – this contributes to both security and availability. In terms of score, however, it lags behind in modularity: its monolithic design does not allow components to be customized or replaced as deeply as in OpenStack. Extension is limited to the defined plugin interfaces, and the overall flexibility of the architecture is lower. Overall, CloudStack is suitable for medium-sized multi-tenant clouds where robust functionality is required without complicated administration; however, advanced users may run up against the limits of the architecture's capabilities when they have very specific needs..

Score: 3 pts – Partial modularity

3.1.2.3 Proxmox VE

Description: Compared with cloud platforms, Proxmox VE is built on a more monolithic architecture. It is an integrated solution for the management of virtualization (KVM), containers (LXC), storage (Ceph, ZFS) and networking, with its own web interface that manages the basic services (virtual machines, storage, network). Clustering functions are built in. Although it uses open-source components (KVM, ZFS, Corosync for the cluster), these are not primarily designed as separate services with their own public APIs that would allow them to be deployed

or replaced independently. Extensibility is achieved rather through the integration of storage backends or by means of scripts and direct access to Linux components [19]. It provides

- **Modularity**: It is based on a monolithic design in which many functions are integrated into a single node, although it supports clustering for high availability. This design limits extensibility and scalability.
- **Microservices**: It does not support a microservice architecture; it is more of an integrated platform, which increases the risk of an outage of the entire system if a node fails.
- **Separately deployable components**: No, many functions are tied to the node, which limits independent deployment and tuning.
- **Plugin integration**: It supports some plugins, such as software-defined storage and networking, but it is not as flexible as OpenStack or Kubernetes [21].
- **Containerization and orchestration**: It supports containers through LXC, but it lacks native orchestration like Kubernetes, which limits its use for modern microservices.

Rationale: From the perspective of criterion K2, Proxmox has the least modular architecture among the compared platforms. Although its "all-in-one" design brings ease of deployment and low complexity (which is positive in terms of administering smaller deployments), it lacks a more pronounced decomposability of components and support for a multi-tenant model. For SOCaaS with strict tenant segregation, Proxmox would have to be supplemented with custom add-on layers; the architecture itself was not primarily designed for such use. An advantage is that the integration of compute, networking and storage in Proxmox is very tight and optimized – for smaller scalable units, this brings efficiency.

Score: 1 pt – monolithic architecture

3.1.2.4 Kubernetes

Description: Kubernetes is designed to support high modularity and is built on a microservice architecture. Its Control Plane consists of components such as the API Server, etcd, the Scheduler and the Controller Manager, each of which runs as a separate service. The Worker Nodes run kubelet and kube-proxy. Kubernetes is extensively extensible through Custom Resource Definitions (CRDs), operators and an extensive ecosystem of plugins (e.g. CNI for networking, CSI for storage) [5]. This architecture is inherently distributed and resilient [20]. It provides:

- **Modularity**: It is highly modular, with components such as the API Server, the Controller Manager and the Scheduler, which can be deployed separately. This design supports extensibility and scalability, which is key for the INNOSOC project.
- **Microservices**: It is designed for the orchestration of microservices in containers, which reduces the risk of an outage when a single service fails.
- **Separately deployable components**: Yes, each component can be deployed and managed separately, which enables independent tuning and replacement of modules.
- **Plugin integration**: It supports a wide range of plugins through interfaces such as CNI (Container Network Interface) and CSI (Container Storage Interface), which increases customizability.
- **Containerization and orchestration**: It is a state-of-the-art container orchestrator, supporting scaling, rolling updates and other modern features, which is in line with the project requirements.

Rationale: Kubernetes is an example of a modern and highly modular architecture based on microservices. Its components are separate, distributed and extensively extensible through CRDs and plugins. Although it is not primarily an IaaS platform for VMs, the architecture itself is outstandingly modular and resilient, which is in line with the requirements of the criterion. A major benefit is the shared ecosystem with cloud-native technologies –

security monitoring tools, CI/CD pipelines and logging can all be unified for both containers and VMs. Although multi-tenancy is not entirely effortless, the platform provides sufficient means to achieve it. Compared with traditional cloud platforms, we see lower vertical complexity here (fewer heterogeneous services than OpenStack) and better horizontal scalability (easy addition of nodes and controllers). For criterion K2, Kubernetes+KubeVirt obtains the highest score, because it fulfils microservice modularity, extensibility and the future-proofness of the architecture for future needs. From the SOCaaS perspective, this means a platform on which multi-tenant security services can be built using the latest trends in automation and orchestration

Score: 5 pts

3.1.3 K3: Support for orchestration and automation (weight: 4)

Rationale: The platform should support tools such as Heat (OpenStack orchestration), Terraform (infrastructure as code), Ansible (automation) or Kubernetes Operator for the efficient management and deployment of complex infrastructure scenarios. This capability will enable rapid scaling of services, automated roll-out of environments, repeatable testing and recovery from outages. In a SOC environment, where requirements and network topology change frequently, orchestration is essential to ensure flexibility, sustainability and operational efficiency.

Scoring recommendation:

- 5 points: Excellent support (Heat, Terraform, Ansible, Kubernetes), robust and flexible automation.
- 3 points: Basic orchestration available, partial limitations in tool support.
- 1 point: Very limited or no automation, weak integration with popular tools.

3.1.3.1 OpenStack

Description: OpenStack is designed with automation and orchestration in mind, which is key to building dynamic cloud environments:

- **Heat:** It is the native OpenStack orchestrator, which makes it possible to define complex cloud environments as code (Infrastructure as Code – IaC) using Heat Orchestration Templates (HOT) [21]. It supports the deployment and management of entire application "stacks", including VMs, networks, storage, security groups and auto-scaling groups.
- **Terraform:** OpenStack has an official provider for HashiCorp Terraform [22], which makes it possible to manage OpenStack resources using declarative Terraform code. It is a very popular tool for multi-cloud and hybrid-cloud IaC. Fully compatible, it enables advanced automation.
- **Ansible:** OpenStack has extensive Ansible modules for automating administrative tasks and the deployment of applications on VMs in OpenStack [23]. Many OpenStack deployments (e.g. Kolla-Ansible) themselves use Ansible for installation and configuration.
- **Kubernetes Operators:** For managing OpenStack running on Kubernetes (e.g. via OpenStack-Helm or other projects), Kubernetes Operators can be used for the automated deployment and management of OpenStack components.
- **Ceilometer/AODH:** OpenStack also has built-in mechanisms for monitoring (Ceilometer) and auto-scaling (AODH - Alarm Orchestration and Decision Engine), which enable automatic adjustment of resources based on load.

Rationale: OpenStack is built on an API-first approach, which makes it an ideal platform for automation. With the native Heat orchestrator, robust integration with Terraform and extensive Ansible modules, it offers excellent support for IaC and the automation of complex scenarios, which is key for the dynamic SOCaaS environment.

Score: 5 pts

3.1.3.2 CloudStack

Description: CloudStack also places emphasis on automation and has a well-defined API.

- **API:** CloudStack provides an extensive RESTful API that makes it possible to manage all aspects of IaaS (VMs, networks, storage, etc.) programmatically [24]. This is the basis for further automation.
- **Terraform:** There is an official Terraform provider for CloudStack [25], which makes it possible to define and manage CloudStack infrastructure using IaC.
- **Ansible:** Like OpenStack, CloudStack is also supported through Ansible modules, which enable the automation of CloudStack administration tasks and application deployment [26]. The available modules are less developed.
- **CloudStack Orchestration:** CloudStack does not have a direct equivalent of Heat Orchestration as in OpenStack, but its Service Offerings and Network Offerings, together with the API, enable a certain degree of orchestration. For more complex scenarios, it usually relies on external tools (such as Terraform or custom scripts/applications calling the API).

Rationale: CloudStack provides a robust API and good integration with popular IaC and automation tools such as Terraform and Ansible. Although it lacks a native comprehensive Heat-type orchestrator, programmatic management of its resources through the API is possible.

Score: 4 pts

3.1.3.3 Proxmox VE

Description: Proxmox VE offers an API and automation tools, but it is primarily designed for server-level virtualization management, not as fully fledged cloud orchestration:

- **REST API:** Proxmox VE has a well-documented RESTful API that makes it possible to automate most VM, container, storage and cluster management tasks [27]. This is the basis for scripting and integration.
- **Terraform:** There is an unofficial but popular Terraform provider for Proxmox VE [28], which makes it possible to manage VMs, storage and networks using IaC.
- **Ansible:** Proxmox has extensive support through Ansible modules [29], which make it possible to automate VM deployment and configuration as well as Proxmox cluster management. The available modules are less mature.
- **Lack of native cloud orchestration:** Proxmox has no built-in orchestrator for complex cloud scenarios (e.g. automatic application scaling, management of "stacks" of multiple interconnected components with complex dependencies) such as Heat in OpenStack. Such functions would have to be implemented through external tools and scripting based on its API.

Justification: Proxmox VE has a robust API and very good integration with popular automation tools such as Terraform and Ansible for managing virtual resources. However, it lacks a native orchestrator for complex cloud scenarios and automatic scaling of services in a

multi-tenant environment. Orchestration would be possible, but it would require significant additional effort through external tools and scripting.

Score: 3 pts

3.1.3.4 Kubernetes

Description: Kubernetes is very strong in orchestration and automation, albeit primarily for containers:

- **Native orchestration:** Kubernetes itself is a container orchestrator. It manages the deployment, scaling, self-healing and life-cycle management of containerized applications. Concepts such as Deployments, StatefulSets, Services and Horizontal Pod Autoscalers form its core [30].
- **Kubernetes Operators:** A key mechanism for automating complex applications. Operators are software agents that automate the deployment and management of applications in Kubernetes, which is ideal for SOCaaS services [31].
- **Infrastructure as Code (IaC):** Everything in Kubernetes is defined declaratively using YAML files, which is a natural form of IaC.
- **Terraform:** Kubernetes has an official provider for Terraform [32], which makes it possible to manage Kubernetes resources using Terraform.
- **Ansible:** K8s also has rich support through Ansible modules for interacting with a Kubernetes cluster and managing its resources [33].
- **KubeVirt:** For the automation and orchestration of VMs on Kubernetes (as containers), KubeVirt is used, which leverages Kubernetes orchestration.

Justification: Kubernetes is a leading platform for the orchestration and automation of containerized workloads. Although VMs are not its primary focus, its fundamental principles (declarative configuration, operators, IaC) and extensive support for tools such as Terraform and Ansible make it an extremely flexible platform for automating complex scenarios, even for VMs via KubeVirt.

Score: 5 pts

3.2 TECHNICAL CRITERIA

3.2.1 K4: Performance and scalability (weight: 5)

Justification: The evaluation considers how efficiently the solution can utilize resources, what limits it has in terms of the number of VMs, containers or tenants, and which automatic scaling mechanisms it supports. Latency and overall performance in data processing are also important, especially in real time for the purposes of security event monitoring. The solution must be able to grow with the number of VMs, tenants and services. Horizontal and vertical scalability are evaluated. Under the conditions of the project, we need a platform that can grow together with the number of SOC service clients and the volume of data processed.

This criterion is therefore assigned a **weight of 5** – maximum importance, since insufficient performance or limited scaling would represent a fundamental risk.

Scoring recommendation:

- 5 points: Excellent scalability, proven ability to handle hundreds of VMs, horizontal and vertical scaling.

- 3 points: Adequate scalability with certain limitations (tens of VMs without a significant drop in performance).
- 1 point: Poor performance, significant scaling limits even with a low number of VMs.

3.2.1.1 OpenStack

Description: OpenStack is a highly scalable solution designed for large deployments, handling from a few up to thousands of virtual machines. It supports both horizontal and vertical scaling of components (Nova, Neutron, Cinder):

- The Nova scheduler can efficiently distribute the load across hundreds of compute nodes, and the system has built-in mechanisms for automatic load balancing and VM migration in the event of performance problems.
- **Horizontal scalability:** Each OpenStack component (Nova, Neutron, Cinder, etc.) can be scaled horizontally by adding further service instances on Compute nodes. This makes it possible to serve hundreds to thousands of hypervisors and thousands of virtual machines [36] [37].
- **Vertical scalability:** Support for various VM types ("flavors") makes it possible to allocate more CPU, RAM and disk space to individual VMs.
- **Horizontal vs. vertical scaling:** OpenStack primarily scales horizontally – by adding further compute nodes to the cluster (or by dividing it into multiple "cells" or regional clusters in very large deployments). At the same time, it also allows **vertical VM scaling** – the administrator or the platform itself can change the flavor (size) of a running VM. Support for **hot-plug** addition of CPU and RAM has historically been limited, but with KVM and modern operating systems it is possible in OpenStack to dynamically add vCPUs or memory to a running VM (this requires support both in the hypervisor and inside the guest). OpenStack can thus also respond to changing application demands without the need for a restart (e.g. by increasing the RAM of a database server at runtime).
- **Resource utilization:** OpenStack seeks to utilize the underlying physical resources efficiently through advanced VM scheduling based on resource availability and defined policies.
- **Latency and real time**: For low latency and near real-time data processing (important for a SOC), OpenStack offers options such as SR-IOV and DPDK for networking, which allow VMs direct access to the hardware, bypassing the hypervisor's virtual network layer [3]. This significantly reduces latency when processing high volumes of network traffic.
- **Automatic scaling:** Using Heat (orchestration) and Ceilometer/AODH (monitoring and alarms), it is possible to implement automatic VM scaling based on load metrics or defined policies, which ensures a dynamic response to load changes.
- **VM management performance:** In OpenStack, each VM runs directly on the hypervisor (typically KVM) with near-native performance. The overhead of OpenStack manifests itself rather in the control plane (API, scheduler, network and storage services). With large numbers of VMs, load distribution is important – OpenStack makes it possible to scale the control services by deploying multiple instances (e.g. parallel API and scheduler nodes). Tests have shown that, for example, the **Nova Scheduler** service should be run in multiple threads/processes with hundreds of nodes (in one test, 8 scheduler processes ran for 1000 nodes). The Neutron network service supports distributed routing (DVR) to reduce latency and load in network communication across nodes.
- **Latency and limits:** As regards latency, with the correct configuration OpenStack maintains reasonable VM creation times even at larger scale. For example, Red Hat

states that by improving database and scheduler performance it is possible to maintain acceptable VM start-up times even with ~1000 nodes. In any case, at extreme scale (hundreds of nodes) a certain delay in some operations must be expected (e.g. network initialization for a VM may take a few seconds). Typical limitations are determined by the configuration: default settings (e.g. the number of Heat engine threads, the queue size in RabbitMQ) may need to be increased for smooth operation in large clusters. The official documentation recommends that, with ~1000 nodes, separate control nodes be dedicated to key services (Nova, Neutron, Keystone, etc.) in order to avoid bottlenecks.

According to tests from practice and academic sources (IEEE), OpenStack is capable of scaling to hundreds or even thousands of VMs without problems. OS provides

- Scalability: It supports horizontal scaling by adding more nodes and vertical scaling by increasing resources on existing nodes. The documentation indicates that it can manage thousands of VMs and compute nodes, with tests on 1,000 nodes and 10–30 thousand VMs [36].
- Performance: Optimized for performance, with features such as live migration, load balancing and efficient resource allocation. It is suitable for real workloads, including security event monitoring, with low latency and high throughput.
- Real-world use: Used by large organizations such as telecommunications companies, governments and enterprises, which demonstrates its ability to handle large infrastructures.

Justification: OpenStack is designed for extremely high scalability, both horizontal and vertical. Its modular architecture allows individual services to be scaled, and advanced networking techniques ensure low latency. It is demonstrably capable of handling hundreds to thousands of VMs in production environments, which fully meets the requirements for SOCaaS growth.

Rating: 5 pts – Excellent scalability, proven ability to handle hundreds of VMs, with support for horizontal and vertical scaling.

3.2.1.2 CloudStack

Description: CloudStack is also known for its robust scalability and is optimized for managing large numbers of VMs. The advantage of CloudStack is that the simplicity of its architecture initially brings low overhead (quick cloud instantiation for smaller deployments); however, at extreme scale the monolith may hit a performance ceiling – at that point it is necessary to scale vertically (more powerful HW for management and the DB, or clustering them). As a result, CloudStack handles common large deployments excellently, but above a certain threshold (on the order of thousands of hosts) it might require architectural modifications or division into several independent CloudStack instances:

- The **scalability** of CloudStack has been proven in many production deployments of hosting and cloud providers; although it is not deployed as often in truly massive clouds as OpenStack, it can handle hundreds to the low thousands of physical hosts. CloudStack uses a zone-based architecture – the infrastructure is divided into zones, pods and clusters. Each cluster usually groups several hosts (hypervisors) with shared storage. Scaling is carried out by adding further clusters and zones, which are still managed by a central management server (or several in HA mode). The advantage is that the load is thereby distributed to a certain extent (e.g. database load by zone).

- **Horizontal scalability:** The CloudStack architecture is designed for simple horizontal scaling by adding further compute nodes (hypervisors) [38]. A single management server or a cluster of management servers can manage thousands of physical hosts and tens of thousands of VMs. It can manage a large number of VMs, but it appears to be less scalable than OpenStack, being better suited to small to medium-sized deployments.
- **Vertical scalability:** It supports dynamic changes of resources (CPU, RAM) for VMs at runtime, if the hypervisor allows it. The administrator can change the "Service Offering" (i.e. the CPU/RAM parameters) assigned to a VM – if the VM has the necessary drivers installed (e.g. VMware Tools for VMware, Xentools for XenServer), CloudStack can increase the vCPUs or RAM of the given VM at runtime. This dynamic CPU/RAM scaling is supported, for example, for VMware and XenServer hypervisors. On the KVM platform, a change of offering is usually possible only after the VM has been shut down (CloudStack restarts the VM with the new parameters). Thus, although CloudStack does not have a fully universal hot-plug capability for all hypervisors, in a VMware/Xen environment it can scale VMs vertically without downtime.
- **Performance**: Good performance for its intended use, but not as optimized for very large or high-performance environments. Research covers parameters such as CPU, memory, disk I/O and network performance, but with fewer references for large scales [39].
- **Resource utilization:** Efficient resource scheduling and allocation, with the option of setting quotas for users and domains.
- **Latency and real time:** Depends primarily on the underlying hypervisor and network infrastructure. CloudStack does not directly offer network path optimizations as advanced as SR-IOV in OpenStack as a built-in cloud function, but it allows them to be configured at hypervisor level.
- **Automatic scaling:** CloudStack has built-in automatic scaling functions (AutoScale), which make it possible to add or remove VMs based on defined metrics [5].
- **Real-world use:** Used by a number of cloud providers and enterprises, but its adoption is not as wide as that of OpenStack, which may indicate scalability limitations for the largest deployments.

Justification: Apache CloudStack offers very good performance and scalability for most typical deployments. Its integrated nature simplifies management and achieves efficiency in small and medium-sized clusters, while handling hundreds of nodes and thousands of VMs without problems. The central management can be regarded as the greatest limitation – in an extremely large infrastructure it becomes a bottleneck, which reduces scaling flexibility compared with the more distributed architecture of OpenStack. OpenStack is therefore often preferred in environments where scaling to thousands of nodes is planned, although through improvements (caching, scheduler optimizations) CloudStack has managed to push its limit significantly and also works reliably in fairly large clouds.

Score: 4 pts

3.2.1.3 Proxmox VE

Description: Proxmox VE is scalable, but primarily within a cluster and for direct VM management:

- **Performance:** In terms of the performance of a single VM, Proxmox uses proven technologies – the KVM hypervisor for full virtualization and LXC for containers – so the performance of running VMs is near-native (KVM is type 1 with minimal overhead). Proxmox adds a thin management layer (web interface, API), which in itself does not

represent significant performance overhead. This means that, compared with other platforms, Proxmox will not have worse raw VM performance – in fact, VMs in Proxmox achieve the same performance as VMs in OpenStack or CloudStack on comparable hardware, since all of them use KVM/QEMU. The I/O and network communication latency of VMs is minimized by using virtio drivers, and Proxmox also supports techniques such as PCI passthrough or SR-IOV for a possible performance increase (e.g. for network cards).

- **Performance limits:** With ~30 nodes in a cluster, Proxmox can safely run hundreds of VMs (the number of VMs is not directly limited by the cluster, but rather by the hardware of each node and the capacity of the shared storage). In the pmxcfs (Proxmox Cluster File System) configuration, it is stated that the size of the config database is limited to 30 MB in RAM, which is sufficient for the **configuration of thousands of VMs**. The KVM hypervisor itself can host tens of VMs on a single node (depending on CPU/RAM performance), so from the point of view of **compute capacity** Proxmox is not an obstacle – you can have, for example, 5 nodes, each with 50 VMs, a 250-VM cluster in total, without problems. However, a bottleneck would arise if you wanted to have 250 VMs *on each node* and, in addition, many nodes – in that case the management (web GUI, API) could respond more slowly, because it has to aggregate the state of hundreds of VMs from many nodes.
- **Horizontal scalability:** Proxmox VE supports clustering, which makes it possible to manage multiple physical nodes and distribute VMs among them. A Proxmox VE cluster can efficiently manage tens of physical nodes and hundreds of virtual machines without a significant drop in performance [39] [42]. Larger clusters are possible, but they require more precise planning and storage management.
- **Vertical scalability:** VMs in Proxmox VE can be flexibly configured with different amounts of CPU, RAM and disk space, including adding resources at runtime.
- **Scalability:** Designed for small to medium-sized environments, such as data centres or enterprise settings, not for large cloud deployments. It can manage a considerable number of VMs, but it is not typically used for hundreds or thousands of VMs across multiple data centres. A forum indicates that a native tool for managing multiple clusters is lacking, which limits scalability [41].
- **Resource utilization:** Proxmox VE makes efficient use of the KVM hypervisor and LXC containers, which ensures good performance.
- **Latency:** VM latency depends on the underlying hardware and the network configuration. Proxmox does not directly provide cloud-native mechanisms for latency optimization (such as SR-IOV/DPDK abstractions), but these technologies are supported at the level of the Linux kernel that Proxmox uses. Their configuration, however, would be "outside" the Proxmox GUI.
- **Automatic scaling:** Proxmox VE has no built-in functions for automatic VM scaling based on load. This would have to be implemented using external scripts or tools that would call the Proxmox API.
- **Real-world use:** Popular among smaller organizations for backup and disaster recovery, but not commonly used for large cloud services or SOC operations.

Justification: Proxmox VE is capable of managing tens to hundreds of VMs in a cluster environment with good resource utilization efficiency. It has good vertical VM scalability. However, it lacks native cloud-level automatic scaling mechanisms and advanced integration for latency optimization at IaaS level. Proxmox VE delivers reliable virtual machine performance and is sufficiently scalable for small and medium-sized clouds; however, compared with the other evaluated platforms, it falls behind in very large deployments. A positive aspect is the almost zero overhead beyond KVM (VMs run at native speed) and the simplicity of clustering on a smaller scale. On the other hand, scalability is limited – Proxmox cannot grow linearly to hundreds of nodes because of the limitations of its cluster stack. It also

lacks native mechanisms for automatic scaling and advanced workload scheduling, which are in demand in dynamic cloud environments. In summary, Proxmox is excellently optimized for simplicity and performance on a smaller scale, but for large multi-tenant clouds it would be necessary to turn to more robust solutions.

Score: 3 pts

3.2.1.4 Kubernetes

Description: Kubernetes is an extremely scalable platform, albeit primarily for containers. Its performance and scalability are among its greatest strengths:

- **Horizontal scalability:** Kubernetes is designed for horizontal scaling of pods (containers) and nodes (physical/virtual servers) to tens of thousands of nodes and hundreds of thousands of pods [40]. This makes it possible to serve enormous numbers of concurrent workloads.
- **Vertical scalability:** It supports vertical scaling for containers through the configuration of CPU/RAM limits.
- **Resource utilization:** It uses resources efficiently thanks to advanced scheduling (scheduler) and automatic recovery.
- **Performance**: Optimized for performance, with features such as auto-scaling, load balancing and efficient resource management. It is suitable for real workloads, including security event monitoring, with blogs on performance improvements since version 1.2
- **Latency and real time:** For containers, Kubernetes is capable of achieving low latency and high throughput. Support for SR-IOV and DPDK is also possible in containers through specific CNI plugins.
- **Automatic scaling:** It has native and very robust automatic scaling mechanisms: Horizontal Pod Autoscaler (HPA), Vertical Pod Autoscaler (VPA) and Cluster Autoscaler [41]. This is key for dynamic environments such as SOCaaS.
- **KubeVirt for VMs:** When KubeVirt is used, VMs run in containers and can use the Kubernetes scaling mechanisms. However, the performance of a VM in a container may be affected by the additional layer.
- **Performance tests and limits:** Several tests have been carried out to verify the scalability of KubeVirt. Of interest was a Red Hat test in which they tried to increase VM density per node – they achieved **400 VMs (VMIs) on a single node** with 128 GB of RAM. It was necessary to raise the default limits (Kubernetes allows 110 pods per node by default; in the test this was increased to ~1022) and to tune the parameters of the kubelet and the KubeVirt controllers (e.g. increasing the API request limits from 5 to 50 per second for the KubeVirt controller). The results showed that at such extreme density latencies already increase: **100 VMs** per node were started in ~3.3 minutes, **200 VMs** in ~10 minutes, and with 300–400 VMs the time stabilized at around 10+ minutes. These figures reflect the 95th percentile of the time from the request to a fully running VM. The limiting factor became the number of container creations per minute – saturation occurred at around **48 VMs per minute per node**. At the same time, this confirmed that scaling to 500 VMs per node was no longer possible due to resource exhaustion (the container runtime started to fail when attempting to start more). This test, however, is an extreme case intended rather for verifying the control plane; in practice, VMs would be distributed across many nodes, so in reality **KubeVirt can handle hundreds to thousands of VMs in parallel** provided they are distributed. The Kubernetes etcd database can sustain the aforementioned ~150k objects, so even if each VM had 2–3 objects (VM, Pod, Service), a cluster with 5000 nodes would be able to manage on the order of tens of thousands of VMs.

- **Real-world use:** Widely used by large cloud providers and enterprises for managing containerized applications, including SOC operations, where real-time monitoring and response are critical. Google Kubernetes Engine (GKE), Amazon EKS and Azure AKS are examples of massively scalable production deployments of Kubernetes.

Justification: Kubernetes excels in scalability, above all for containerized workloads. It offers native and robust mechanisms for both horizontal and vertical automatic scaling and is designed for enormous scales. Although VMs are not its primary focus, its fundamental properties and ability to scale workloads (including VMs via KubeVirt) make it extremely suitable for the dynamic and growing requirements of SOCaaS.

Score: 5 pts

3.2.2 K5: Reliability and stability (weight: 4)

Justification: We assess the degree of maturity of the platform, its stability in production operation and the availability of mechanisms for high-availability deployment. The criterion covers the level of testing and debugging of the software and the possibility of deploying components redundantly (HA – High Availability). In a security centre environment, it is important to minimize downtime and ensure continuous availability of the service. When evaluating cloud services, it is recommended to examine thoroughly past performance, customer satisfaction and the handling of unplanned outages. We also compare SLAs (Service Level Agreements) where available – although in the case of an open-source platform these are rather internal guarantees, availability of support, etc.

Since reliability directly affects the quality of the SOC service provided, this criterion has a **weight of 4** (higher importance).

Scoring recommendation:

- 5 points: No serious outages, reliable production references, support for HA deployment.
- 3 points: Occasional minor outages, partial HA support.
- 1 point: Frequent or significant outages, no HA support.

3.2.2.1 OpenStack

Description: OpenStack is a long-developed platform that has been under development for more than 14 years and is extensively deployed in critical production environments around the world [46]. Reliability and high availability are fundamental pillars of its design. Today it is one of the most widely used open-source cloud platforms: according to surveys, over 40 million VM cores run in production OpenStack clouds around the world [47]. References confirm its reliability – OpenStack is used, for example, by AT&T for its mobile core network, Walmart operates almost a million cores, and China Mobile is building an NFV network with 50 thousand servers. Companies such as Blizzard (gaming infrastructure) and PayPal (over a billion transactions a day) also use OS. Large scientific and academic clusters (CERN, NSF JetStream) have also confirmed the stability and reliability of OpenStack at large scale:

- **Maturity and testing:** Thanks to its huge global community and extensive production deployments (including the data centres of large enterprises and telecommunications operators), OpenStack is thoroughly tested and debugged. Each new release goes through rigorous testing cycles [46].
- **High availability (HA):** OpenStack supports deployment in HA mode at several levels. The control services (control nodes) can be operated redundantly (typically 3+

controllers) using clustering technologies such as Pacemaker and the HAProxy load balancer, thereby eliminating any single point of failure. The control plane can thus run in *active/active* mode (several instances of a service at once) or *active/passive* mode, depending on the nature of the component . The database layer uses a replication cluster (e.g. Galera for MariaDB) for resilience against the failure of a DB node. Compute nodes: hypervisor failures as such do not affect the operation of other VMs; OpenStack has mechanisms for automatic VM recovery. The Masakari component provides *high availability* for instances by automatically restarting the affected VMs on live nodes when a hypervisor fails.

- **Support for HA deployment:** All key OpenStack components (Nova API, Neutron API, Cinder API, Keystone, Glance, etc.) can be deployed redundantly in a high-availability (HA) configuration. This is typically achieved by means of:
    - Multiple instances of control services behind a load balancer.
    - Cluster solutions for databases (e.g. Galera Cluster for MySQL) and message handling (e.g. RabbitMQ cluster).
    - Distributed file systems and storage solutions (Ceph, Swift) for resilience against storage failure.
    - Mechanisms for preventing a "single point of failure" at the network and compute levels.

Justification: OpenStack is a highly mature platform with proven stability in large-scale production deployments. It offers comprehensive and robust mechanisms for implementing high availability for all its components, minimizing downtime and ensuring continuous availability of the SOCaaS service.

Score: 5 pts

3.2.2.2 CloudStack

Description: CloudStack is also an established platform that has become a reliable IaaS solution, particularly in service provider environments:

- **Maturity and testing:** CloudStack is an open-source project under the auspices of the Apache Software Foundation with a long development history and extensive production use. This ensures a high level of testing and debugging [48].
- **Support for HA deployment:** The CloudStack management server (which is a potential single point of failure) can be deployed in high-availability (HA) mode with an active-passive or active-active configuration [49]. It also supports HA for databases and network services. The VMs on the hypervisors themselves are protected by the hypervisor's high-availability mechanisms (e.g. VMware HA, or KVM with Pacemaker/Corosync).
- **Production references:** Many cloud providers and large enterprises use CloudStack for critical services, which testifies to its stability and reliability in production environments (e.g. T-Mobile, NTT Communications, Citrix). Although not as widely publicized as OpenStack, its references are solid [50]

Justification: CloudStack is a mature and reliable platform with proven stability in production environments. It offers robust mechanisms for high availability of the management server and supports HA functions for hosts, which ensures continuous availability of SOCaaS services.

Score: 5 pts

3.2.2.3 Proxmox VE

Description: Proxmox VE is a stable and robust virtualization platform that provides cluster-level high availability for virtual machines:

- **Maturity and testing:** Proxmox VE has an active community and a stable development cycle with regular updates, which indicates a good level of software testing [51]. It is widely used in small and medium-sized companies as well as in the academic sector.
- **Support for HA deployment:** Proxmox VE has a built-in high-availability (HA) cluster based on Corosync and Pacemaker, which enables automatic migration of virtual machines and containers to another functional node in the event of hardware failure [52]. This ensures high availability of the VMs themselves. The management layer (web interface and API) is distributed within the cluster and, if one node fails, it is available via another.
- **Production references:** Proxmox is reliable for deployments ranging from tens to hundreds of VMs. Although it may not have references from giant companies like OpenStack, its stability in real production environments is generally recognized within its target segment.

Justification: Proxmox VE is a stable platform with robust built-in mechanisms for high availability of VMs in a cluster. Although it is a virtualization solution rather than a fully fledged cloud platform, its ability to ensure the continuous operation of virtual components for SOCaaS is very good and has been proven in many production environments.

Score: 4.5 pts

3.2.2.4 Kubernernetes

Description: Kubernetes has been developed from the ground up for high availability, fault tolerance and self-healing capabilities, which is critical for microservice architectures and dynamic environments:

- **Maturity and testing:** Kubernetes is the de facto standard for container orchestration, used by thousands of companies around the world, including the very largest. It is intensively tested and continuously improved by a huge community and by companies such as Google, Red Hat, etc. [53].
- **Support for HA deployment:** The entire Kubernetes control plane can be deployed in a highly available configuration [10], in which components such as the API Server, etcd, Scheduler and Controller Manager run redundantly. Worker nodes are also redundant, and Kubernetes automatically reschedules the pods of failed nodes onto healthy ones. Built-in mechanisms such as "self-healing" (restarting crashed containers) and "liveness/readiness probes" ensure continuous availability of applications.
- **Production references:** Millions of applications run on Kubernetes in critical production environments with extremely high SLA requirements (e.g. Spotify, Airbnb, IBM, Google Cloud, AWS, Azure) [53]. These references confirm its excellent reliability and stability.
- **KubeVirt stability:** When KubeVirt is used for VMs, VM stability is affected by the stability of the Kubernetes cluster and of KubeVirt itself. Since KubeVirt uses native Kubernetes mechanisms, it benefits from its HA capabilities.

Justification: Kubernetes is designed from the ground up for extremely high reliability and stability. Its architectural principles include control plane redundancy, self-healing capabilities and robust mechanisms for automatic workload recovery. It is the de facto standard

for critical production deployments, which ensures continuous availability of SOCaaS components.

Score: 5 pts

3.2.3 K6: Compliance with standards and interoperability (weight: 4)

Justification: From a technical point of view, it is significant whether the platform supports open standards and common technologies, which determines its interoperability with other systems. The solution should be compatible with popular hypervisors, network protocols, storage technologies and management interfaces (e.g. the OpenStack API is the de facto standard for open-source clouds). Broad support for standards also facilitates integration and minimizes the risk of proprietary limitations. We also assess the availability of certifications or references – for example, whether the platform is based on proven open-source projects supported by a large community. As cloud provider evaluation practices indicate, adherence to industry standards and certifications (e.g. ISO 27001 in security) is a significant selection criterion. In our project, interoperability will make it possible in future to interconnect multiple cloud environments and integrate further components.

It is also important that the platform contains tools for implementing the required measures (e.g. logging, encryption, access controls) and that the community/vendors provide guidance on regulatory compliance. Regulatory compliance will affect the acceptability of the solution in production – for example, for public administration data the platform should support deployment in compliance with the personal data protection law (GDPR). Given the nature of the planned SOC solution (which may process clients' sensitive security data), we consider this criterion to be very significant and assign it a weight of 4.

Scoring recommendation:

- **5 points:** Full compatibility with open standards, certifications (ISO 27001, GDPR compliant), active interoperability.
- **3 points:** Partial compatibility and interoperability, some standards are not fully covered.
- **1 point:** Limited or no compatibility, interoperability problems.

3.2.3.1 OpenStack

Description: OpenStack is an open-source IaaS platform with very good support for standards and interoperability. OpenStack is designed as an open cloud platform with an emphasis on adherence to industry standards. It supports multiple hypervisors (KVM, Xen, VMware, Hyper-V) and uses proven open-source technologies such as libvirt and Open vSwitch. Its APIs have become the de facto standard for open-source clouds – for example, many third-party tools support the OpenStack API, and OpenStack itself uses well-known formats (e.g. S3 compatibility for object storage, or compatibility with CloudFormation templates in orchestration) [56]. Thanks to broad support for standard protocols (HTTP/REST API, POSIX for images, etc.) and open development, the integration of OpenStack with existing systems and services is seamless. It provides:

- **Compliance with standards**: OpenStack has extensive documentation on compliance with standards, including ISO 27001, SOC 1/2 and GDPR (OpenStack Compliance Overview). Its Keystone identity service supports strong authentication and access policies that help meet regulatory requirements such as PCI-DSS (Red Hat OpenStack PCI-DSS).

- **Interoperability**: It supports a wide range of hypervisors (e.g. KVM, Xen, VMware), storage systems (e.g. Ceph, Cinder) and network technologies (e.g. Neutron). The OpenStack API is regarded as the de facto standard in open-source clouds, which facilitates integration with other systems (OpenStack Security Guide).
- **Certifications and community**: OpenStack has a large community and regular updates, which support compliance with standards and provide guidance on security and regulation (OpenStack Cloud Security).

Justification: Fully meets the requirements for compliance with open standards and interoperability. The OpenStack API is a widely adopted standard. It has a robust ecosystem of tools and support, which are essential for achieving and demonstrating compliance with international certifications (ISO 27001) and regulatory requirements (GDPR) for a multi-tenant environment processing sensitive security data. OpenStack has one of the largest open-source communities and is regarded as the global standard for IaaS clouds – it is deployed both in large enterprises and in the telecommunications sector. It respects the principles of interoperability: it reuses existing interfaces and formats (e.g. the AWS S3 API within Swift), and its own component APIs are widely supported in tools and libraries. Independent interoperability tests (the OpenInfra Foundation Interop programme) guarantee that different OpenStack distributions meet uniform requirements, which facilitates the transfer of applications between environments. OpenStack thus minimizes the risk of vendor lock-in and maximizes compliance with common cloud industry standards

Score: 5 pts

3.2.3.2 CloudStack

Description: CloudStack also supports an extensive set of virtualization technologies (KVM, VMware ESXi, XenServer) and provides a consistent API for infrastructure management. Its API is also RESTful and well documented, which enables integration with other systems. The CloudStack community is smaller than that of OpenStack, but it is still active and commercial support is available. In terms of ease of deployment and operation for small and medium-sized environments, it is often preferred. As regards regulatory compliance, CloudStack offers basic functions such as RBAC and network isolation, but the depth and scope of tools, guidance and certifications specific to compliance with strict requirements (such as detailed audits for ISO 27001 or specific GDPR requirements for public administration) may be less developed than for OpenStack. Achieving full compliance could require more manual work and external tools. It provides:

- **Compliance with standards**: There is the ACCEL certification (Apache CloudStack Certification Exam by LPI-JAPAN), which verifies knowledge and skills in working with CloudStack (ACCEL Certification). However, information on specific standards such as ISO 27001 or GDPR is limited, which indicates partial compatibility.
- **Interoperability**: It supports various hypervisors (e.g. VMware, KVM, XenServer) and storage systems (e.g. NFS, Ceph), but its ecosystem is not as extensive as that of OpenStack (Apache CloudStack).
- **Certifications and community**: CloudStack has a smaller community than OpenStack, but it is actively maintained by the Apache Software Foundation, which ensures a certain level of support and updates.

Justification: CloudStack places emphasis on ease of integration – in its basic installation it provides a unified package of services that uses proven open-source components. Thanks to optional compatibility with the AWS API, it increases interoperability with systems originally intended for the Amazon cloud [57]. The Apache community ensures

transparent development and the absence of vendor lock-in: all users use the same open-source code, which increases the mutual compatibility of deployments. Although the CloudStack community is smaller than that of OpenStack, the platform adheres to the main standards in the field of virtualization and networking. A minor shortcoming may be a narrower ecosystem of add-ons, but within standard functionality (hypervisors, networks, storage) CloudStack does not lag behind.

Rating: 3.5 pts

3.2.3.3 Proxmox VE

Description: Proxmox supports standard virtualization protocols and uses the KVM hypervisor (an integral part of the Linux kernel) and LXC containers, which are among the industry standards in virtualization. It supports common disk image formats (QCOW2, RAW) and standard storage technologies (LVM, ZFS, Ceph). Through its REST API (with JSON format), it enables simple integration with third-party tools

However, it does not offer an extensive cloud API compatible with the OpenStack API or other cloud standards for dynamic, programmable orchestration of cloud services. Its web interface and CLI are the primary management tools. Interoperability with complex cloud environments would require considerable effort and the building of additional integration layers on top. As regards regulatory compliance (ISO 27001, GDPR), Proxmox VE does not provide native tools or certification programmes at the level of the entire cloud platform. Ensuring compliance would largely be the responsibility of the implementer and would require a combination of in-house processes, external tools and careful configuration of the operating system,

It provides:

- **Compliance with standards**: There are no specific certifications such as ISO 27001 or GDPR, but Proxmox can be hardened for security and compliance with standards, as stated on the forum (Proxmox Hardening). Support for CIS Benchmarks indicates potential for compliance (Proxmox CIS Benchmark).
- **Interoperability**: It supports various storage technologies (e.g. Ceph, ZFS, NFS) and network protocols, but it is not designed for large-scale cloud deployments, which limits its interoperability (Proxmox VE Administration).
- **Certifications and community**: Proxmox has an active community, but smaller than that of OpenStack or Kubernetes, with limited guidance on regulations.

Justification: It has limited or no compatibility with open IaaS cloud standards for a multi-tenant environment. It is primarily a virtualization platform based purely on open-source components (Debian, KVM, QEMU, LXC).jProxmox highly compatible with common virtualization standards, not a comprehensive cloud solution. It lacks standardized cloud APIs for large-scale orchestration and native support for complex regulatory requirements (such as ISO 27001 and GDPR at the cloud platform level), which would lead to interoperability and compliance problems in the SOC project.

Score: 3 pts

3.2.3.4 Kubernernetes

Description: Kubernetes is the leading open standard for the orchestration of containers and cloud-native applications. It is built on a modular architecture with a rich and flexible API, which is the *de facto* standard for container platforms. It supports a wide range of open standards (OCI for containers, CNI for networking, CSI for storage) and has an

exceptionally active and extensive global community. Within the Kubernetes ecosystem there are numerous tools and operators that facilitate the implementation of security measures (network segmentation, access controls, encryption) and help achieve regulatory compliance (ISO 27001, GDPR) for containerized applications. Many commercial Kubernetes distributions and managed services are certified. However, although it is possible to manage virtual machines in Kubernetes through projects such as KubeVirt, this is not its primary function and it adds a layer of complexity. For a SOC solution built primarily on IaaS (virtual machines), it would not be the most direct solution in terms of native standards. For containerized SOC components, however, it would be ideal.

It provides:

- **Compliance with standards**: It supports frameworks such as CIS Benchmarks, NIST SP 800-53 and 800-190, which facilitate compliance with regulations such as GDPR (Kubernetes Compliance Frameworks). Pod Security Standards enforce security policies (Kubernetes Pod Security).
- **Interoperability**: It supports various container runtimes (e.g. containerd, CRI-O), storage systems (via CSI) and networking solutions (via CNI), which ensures high interoperability (Kubernetes Security).
- **Certifications and community**: Kubernetes has a large community under the auspices of the CNCF, with regular updates and guidance on regulatory compliance (Kubernetes NIST Guide).

Justification: As an open project, Kubernetes has a huge community and almost universal support – according to CNCF surveys, around 80% of organizations use it in production [58]. This guarantees broad interoperability: applications containerized for Kubernetes can easily be moved between different environments. KubeVirt builds on the same principles, bringing VMs into the Kubernetes world without closed formats. Although Kubernetes itself has no built-in component for VM management, KubeVirt is a CNCF project (release v1.0 in 2023) and follows the best practices of community development. Integration with existing tools (CI/CD, Istio service mesh, etc.) takes place via the open Kubernetes API. However, for the basic IaaS functionalities (management of virtual machines), which are defined for this project as key for a multi-tenant SOC, Kubernetes is an extension (via KubeVirt) rather than a native solution. As a result, some IaaS standards are not directly covered, which leads
to "partial" interoperability in the context of the project's needs. A slight limitation may be the complexity of the solution when trying to achieve full compatibility with all IaaS functions, but in terms of standards and interoperability, Kubernetes + KubeVirt scores highly as a modern, open platform

Score: 4 pts

3.3 OPERATIONAL CRITERIA

3.3.1 K7: Complexity of deployment and management (weight: 3)

Justification: This criterion assesses the effort and resources that will need to be expended on initializing the platform and its subsequent management. It covers the complexity of installation (the number and intricacy of the steps involved in deploying the entire environment), the requirements for staff expertise, the time needed for full commissioning, as well as the clarity of maintenance (updates, upgrades, scaling). For example, some open-source platforms are highly configurable but may be complex to deploy, which requires an

experienced team or the use of automation tools (e.g. Ansible scripts, etc.). From an operational point of view, we also evaluate whether the solution provides functionalities that facilitate management – e.g. self-healing, monitoring, simple cluster expansion, etc. The aim is to choose a platform whose operation will not be a disproportionate burden for the organization.

We give this criterion **a weight of 3** (medium importance), given that some additional effort with an open-source solution is acceptable as long as the other benefits outweigh it.

Scoring recommendation:

- 5 points: Simple deployment, clear documentation, minimal management requirements, strong automation.
- 3 points: Moderately more complex deployment, manageable with the available tools and documentation.
- 1 point: Complicated installation, demanding administration, insufficient documentation.

3.3.1.1 OpenStack

Description: OpenStack deployment is known for its complexity, especially for production environments. It requires in-depth knowledge of the various components (Nova, Neutron, Cinder, Glance, Keystone, etc.), network and storage configuration, and integration with hypervisors. There are many installation methods (TripleO, Kolla-Ansible, OpenStack-Ansible, Packstack) which, although they automate the process, still require considerable expertise and experience for proper planning, deployment and tuning. Maintenance, updates and upgrades are also demanding operations that require careful planning and can be time-consuming. However, OpenStack offers an extensive API for automation, and tools exist for monitoring and partial self-healing. Full-fledged management and scaling require an experienced team of DevOps/cloud engineers. In commercial distributions, management is partially simplified, but the underlying complexity remains.

- **Deployment**: Complicated due to multiple components (Nova, Neutron, Cinder); it requires expertise and automation tools such as Ansible or Terraform. Installation can take days to weeks (OpenStack Deployment Guide).
- **Management**: Demanding due to the need for manual configuration and maintenance, although tools such as Heat and monitoring solutions (e.g. Zabbix) facilitate management. The documentation is extensive but requires technical skills (OpenStack Operations Guide).
- **Automation**: Strong support for automation via Heat, Ansible and Terraform, with features such as self-healing when integrated with external tools.

Justification: Complicated installation and demanding administration for a full-fledged production environment. It requires a high level of expertise and considerable resources for deployment, maintenance and life cycle management, despite the existing automation tools. Compared with the other platforms, OpenStack deployment is considered complex – its flexibility comes at the price of a higher administrative burden. túdies report that reliable operation requires a larger team with in-depth knowledge of the components. On the other hand, for large deployments this investment is justified by the robustness of the resulting cloud.

Rating: 1.5 pts

3.3.1.2 CloudStack

Description: CloudStack provides an **"all-in-one"** package – after installing the management server (Java application + database) and the agents on the hypervisors, the basic environment is ready. The preconfigured stack of services reduces the number of steps needed to get the cloud up and running. Compared with OpenStack, CloudStack has fewer separate components, which simplifies the architecture (e.g. networking and storage functions are integrated directly). Installation typically takes place via packages (RPM/DEB) according to the guide, and for smaller deployments even a single management server can serve the cloud. System management (users, zones, templates) can be conveniently performed via the web interface.

Justification: Compared with OpenStack, CloudStack deployment is considerably less complex – the literature states that CloudStack is **„simple to set up"** and suitable even for smaller teams [59] [60] . The integrated approach (pre-prepared combinations of services) eliminates the need for lengthy configuration interventions and for tuning compatibility between components. Organizations without extensive cloud experience can get CloudStack running faster, which shortens the time to put it into operationopenmetal.io. Lower modularity does limit the possible customizations, but at the same time it means fewer places where a misconfiguration can be made. Overall, managing and updating CloudStack is less demanding, the upgrade cycle is slower (LTS versions) and the documentation covers the most common deployment scenarios.

Score: 4 pts

3.3.1.3 Proxmox VE

Description: Proxmox VE is known for its simple installation. A typical installation from an ISO image is straightforward and fast. It has an intuitive web interface that greatly facilitates the management of virtual machines, containers, storage and networking. Adding further nodes to the cluster and scaling is relatively simple. Clear documentation and processes exist for maintenance and updates. It does not require extensive specialized IT knowledge beyond basic virtualization and networking. Features such as live migration, integrated backup and an HA (High Availability) cluster facilitate operation. Automation is possible via the CLI and API, but it is not as extensive for cloud services as in the case of OpenStack. Nevertheless, it is a very user-friendly solution for the virtualization layer.

- **Deployment**: Very simple, with installation via an ISO image that can be completed within hours. The web interface facilitates configuration (Proxmox Installation).
- **Management**: The intuitive web interface does not require in-depth expertise. Support for updates and scaling is simple, with integrated monitoring (Proxmox VE Administration Guide).
- **Automation**: Support for automation via the REST API and Ansible, although less advanced than OpenStack or Kubernetes.

Justification: Simple deployment from ISO, intuitive web interface and clear documentation for management. Minimal management requirements and simple scaling for the virtualization environment. Strong automation for basic virtualization tasks.

Score: 5 pts

3.3.1.4 Kubernetes

Description: Deploying a Kubernetes cluster, especially in a production environment, can be extremely complex. There are many installation methods (kubeadm, kops, Rancher,

OpenShift, cloud services such as GKE/EKS/AKS), each of which has its own specifics and requires in-depth knowledge of Linux, containers, networking (CNI), storage (CSI) and orchestration. Managing and maintaining a Kubernetes cluster requires specialized engineers with extensive skills in cloud-native technologies. Upgrades can be demanding. On the other hand, Kubernetes is designed for a high level of automation (declarative API, operators), self-healing (restarting failed pods, automatic scaling) and monitoring. Once the cluster is deployed and running, managing applications within it can be efficient, but the initial investment in deploying and operating the underlying infrastructure is considerable.

- **Deployment**: Complicated, especially for HA clusters; it requires expertise and tools such as kubeadm or KubeSpray. Installation can take days (Kubernetes Setup).
- **Management**: Demanding due to the complexity of configuration, but tools such as operators, Helm and monitoring (e.g. Prometheus) facilitate management. The documentation is extensive (Kubernetes Documentation).
- **Automation**: Strong support for self-healing, automatic scaling and monitoring, which reduces the operational burden after deployment.

Justification: Complicated installation and demanding administration for a large-scale production environment. It requires specialized expertise and considerable resources for deployment, maintenance and cluster life cycle management, despite excellent native features for automation and self-healing.

Score: 1 pt

3.3.2 K8: Support, availability of documentation and community (weight: 4)

Justification: The level of support available for a given platform significantly affects its practical usability. For commercial solutions, we consider the quality of the vendor's technical support (24/7 availability, SLAs for incident resolution). For open-source solutions, the size and activity of the community are important – the number of contributors, the frequency of new releases and bug fixes, and the availability of documentation and a knowledge base. A strong community and support from large IT companies are a guarantee of the project's long-term sustainability. The existence of training, certifications or consulting services for the given platform is also evaluated. Experience shows that smaller providers may offer more personal support, while larger players have more extensive support resources, but often at an extra charge. Since in our case we are considering an open-source platform, the criterion focuses on community support and the ecosystem around it. The project's development activity and the frequency of updates are evaluated

We assign it **a weight of 4**, as the availability of expert support and active development are very important for the success of the project.

Scoring recommendation:

- **5 points:** Active community, rich documentation, regular updates and professional support.
- **3 points:** Average community, basic documentation, slower updates.
- **1 point:** Weak or non-existent community, outdated documentation, minimal support.

3.3.2.1 OpenStack

Description: OpenStack has an exceptionally broad and active community – thousands of developers around the world have contributed to the project, and more than 110,000 members in 187 countries are registered under the umbrella of the OpenInfra

Foundation. The project has excellent documentation: the official guides cover installation, architecture and API development (e.g. the detailed **OpenStack Documentation** portal). Books, online courses and certifications (COA – Certified OpenStack Administrator) are also available. Large companies (Red Hat, Canonical, Mirantis) offer paid support and regular releases (OpenStack distributions with long-term support). The community organizes annual summits and user groups where experience is shared. The update frequency is ~2× per year; security updates are released on an ongoing basis. . The availability of training and certifications is very high:

- **Community support**: A large and active community with contributions from companies such as Red Hat, Canonical and SUSE. Support is available via mailing lists, forums and local user groups (OpenStack Community). Conferences such as the OpenStack Summit increase community engagement.
- **Documentation**: Extensive documentation at docs.openstack.org covers installation, management, development and troubleshooting. It includes guides for various Linux distributions and deployment scenarios (OpenStack Installation Guide).
- **Update frequency**: New versions are released every six months, named after Texan towns. Older versions have Extended Maintenance, which ensures long-term support (OpenStack Releases).
- **Support and training**: Certifications (e.g. Certified OpenStack Administrator) and professional support from companies such as Red Hat and Canonical are available (Red Hat OpenStack).

Justification: An active and broad community, rich and continuously updated documentation, regular and frequent updates. Extensive professional support and certifications are available from several major commercial entities, which guarantees the long-term sustainability of the SOC project. The platform has long-term sustainable development and support, which is crucial for critical deployments.

Score: 5 pts

3.3.2.2 CloudStack

Description: CloudStack has an active but smaller community compared with OpenStack. It is an Apache Software Foundation project, which ensures a certain level of transparency and an open governance model. New versions are released fairly regularly, although the frequency may be somewhat lower than for OpenStack. The documentation is good and sufficient for common deployments and management, but it may not be as detailed for edge cases or complex integrations. There are some commercial vendors (e.g. ShapeBlue, StorPool) that provide support, training and consulting, but their ecosystem is less extensive. Overall, the support and community are stable, but not as massive or as well resourced as those of OpenStack or Kubernetes:

- **Community support**: A smaller community than OpenStack, but still active, with contributions from companies such as Citrix and ShapeBlue. Support is available via mailing lists and forums (CloudStack Mailing Lists). Discussions on platforms such as Reddit suggest that the community is engaged, although smaller (CloudStack Reddit Discussion).
- **Documentation**: Comprehensive documentation at docs.cloudstack.apache.org covers installation, management and advanced configurations. It is less extensive than that of OpenStack, but sufficient for most uses (CloudStack Documentation).
- **Update frequency**: Regular releases, with the latest LTS version 4.20.1.0 in 2023. Updates are less frequent than for OpenStack, but still reliable (CloudStack Releases).

- **Support and training**: The ACCEL certification from LPI-JAPAN is available, but professional support is limited to a smaller number of providers ([ACCEL Certification]).

Justification: CoudStack maintains an active but relatively narrowly focused community. The number of contributors and users is not as large as for OpenStack; however, reference users include major companies (Telefónica, BT, universities), which testifies to confidence in the project. The documentation is rated as sufficient for common deployments, although it is not as extensive as for OpenStack. An advantage is the Apache model – transparent project governance and a stability-oriented community. The availability of professional support (e.g. ShapeBlue offers 24/7 support) increases its attractiveness for companies that do not have their own team of experts. Overall, CloudStack has a strong position in terms of community and support,

Score: 3 pts

3.3.2.3 Proxmox VE

Description: Proxmox VE is supported by an active community of administrators, enthusiasts and small companies. The official Proxmox forum has tens of thousands of posts in which developers and experienced users advise newcomers (the forum is considered an excellent source of solutions – *„great place to find an answer"*). The project documentation is available on the wiki (pve.proxmox.com) and includes user guides, frequently asked questions and procedures for advanced settings. Proxmox Server Solutions GmbH (the creator of the project) offers a paid enterprise subscription that guarantees access to stable repositories and technical support. The community publishes many tutorials (YouTube, blogs) focused, for example, on clustering, backup, integration with ZFS, etc. Proxmox VE updates are released regularly – major versions approximately once a year, minor updates on an ongoing basis; the upgrade process is well documented:

- **Community support**: An active community with support via the forum (Proxmox Support Forum). Commercial support is available via a subscription, which increases reliability (Proxmox Subscription).
- **Documentation**: Detailed documentation at pve.proxmox.com covers installation, management and advanced features. It is user-friendly and regularly updated (Proxmox Documentation).
- **Update frequency**: New versions are released every few months, with release 8.2 in 2024. Updates are available via repositories (Proxmox Roadmap).
- **Support and training**: Commercial training and support are available, with positive reviews from users ([Proxmox Reviews]).
- **Score**: 5 points – Active community, rich documentation, regular updates and professional support

Justification: Among open-source projects, Proxmox has a strong and engaged community, centred on the official forum and numerous posts on blogs and Reddit from users in the homelab and SMB sphere. As it is a popular solution for smaller deployments, the community is very practically oriented – solutions to common problems can often be found in discussions. The documentation covers key topics, although compared with large cloud platforms it is less comprehensive (it often relies on knowledge of Linux, on which Proxmox is built). The availability of paid support directly from the creators is a big plus for deployment in the enterprise sphere – for a relatively low annual fee it is possible to have access to experts. Overall, Proxmox achieves a high rating, the only reason for not receiving the full 5 points being the smaller scope of the project (and thus of the community) compared with gigantic cloud solutions

Score: 5 pts

3.3.2.4 Kubernetes

Description: Kubernetes has one of the most dynamic and largest open-source communities in the IT world, governed through the Cloud Native Computing Foundation (CNCF). The project has a huge number of contributors from companies such as Google, Red Hat and Microsoft, as well as independent developers. New versions are released extremely frequently (approximately every 3-4 months), which brings rapid development of features and security fixes. The documentation is extremely extensive and continuously updated, although its complexity may be a challenge for newcomers. There are countless books, courses, certifications (e.g. CKA, CKAD, CKS) and consulting services from virtually all major IT companies. The entire ecosystem around Kubernetes is huge and constantly growing.

Justification: The most active and largest community in the cloud-native field. It has extremely rich documentation, very frequent updates and a wide range of professional support, training and certifications from leading IT companies, which guarantees long-term sustainability and robust support for the SOC project.

Score: 5 pts

3.4 SECURITY CRITERIA

3.4.1 K9: Authentication and access control (weight: 5)

Justification: The platform should include the built-in security features needed to protect the infrastructure and data in the SOC environment. These include, for example, support for multi-level authentication and authorization of users, the possibility of data encryption (in storage, in transit), logging and auditing of events, and automated security policies. For an open-source solution, we assess which security modules it offers and whether it complies with modern security practices (regular security updates, hardening options).

Given the nature of the project (cybersecurity), this is a **critical** criterion with a weight of **5**. A platform that could not secure data and systems at an adequate level cannot be considered suitable.

Scoring recommendation:

- 5 points: Advanced multi-level authentication, robust access control, full support for logging and auditing.
- 3 points: Basic authentication and access control, limited auditing options.
- 1 point: Weak authentication security, minimal auditing options.

3.4.1.1 OpenStack

Description: OpenStack includes the robust identity module **Keystone**, which provides user authentication and authorization of their access to resources. Keystone supports multi-level verification – the basis is a token mechanism (the user logs in with a username/password or another mechanism and obtains a time-limited token for calling services). Integration with external directories such as LDAP/Active Directory [61] is possible, as is federated authentication (SAML, OpenID Connect) for single sign-on. Keystone makes it possible to set up multi-factor authentication (MFA) – e.g. a combination of a password and a one-time TOTP code [62]. Access control is handled by means of role-based access control (RBAC): administrators define roles and assign them to users in projects. OpenStack has predefined

roles (admin, member, reader), and since the Wallaby release a unified default RBAC policy has been available for consistent permissions across services.

- **Multi-level authentication and authorization**: It supports integration with external systems such as LDAP/AD, with RBAC based on roles, groups and users
- **MFA support**: It can be extended via plugins or external services, although it is not natively supported.
- **Data encryption**: It supports TLS for data in transit and can integrate encrypted storage for data at rest.
- **Logging and auditing**: Comprehensive logging of authentication and auditing events, with support for tools such as the ELK stack.
- **Automated security policies**: It allows policies for access control to be defined, which can be automated.
- **Modern security practices**: Regular updates and hardening guides .
- **RBAC**: Fully implemented via Keystone, enabling fine-grained access control.

Justification: It provides advanced multi-level authentication and robust access control with detailed RBAC. It offers full support for logging, auditing and encryption of data at rest and in transit, complies with modern security practices and is actively maintained from a security perspective.

Score: 5 pts

3.4.1.2 CloudStack

Description: CloudStack provides a built-in access control mechanism based on accounts, domains and roles. The system distinguishes several levels of administrators: root admin (the entire cloud), domain-admin (administrator of a specific domain/tenant) and regular users [63]. For each account, multiple users who share resources can be defined. Authentication is by default via username/password, and CloudStack supports integration with external systems: it allows connection to LDAP/AD for user verification and also the configuration of Single Sign-On via SAML 2.0 [63]. Since version 4.11, CloudStack has introduced dynamic roles – the administrator can create custom roles with a specific set of permissions (the roles Admin, User and DomainAdmin are predefined). Two-factor authentication is supported by an optional plugin – TOTP code generation can be enabled for logging in to the GUI [63]. Apache CloudStack offers the following features:

- **Multi-level authentication and authorization**: It supports API keys, LDAP, SAML and OAuth2, with hierarchical access control via roles, accounts, users and domains
- **MFA support**: It explicitly supports 2FA and OAuth2, which increases security.
- **Data encryption**: It supports TLS; encryption of data at rest depends on the underlying storage.
- **Logging and auditing**: It provides logging, but auditing may require additional configuration.
- **Automated security policies**: It allows policies to be defined via the API and console, with limited automation.
- **Modern security practices**: Regular updates; it supports modern standards such as SAML and OAuth2.
- **RBAC**: It implements RBAC via roles and domains, enabling fine-grained control.

Justification: CloudStack's security features cover most of the needs of a multi-tenant cloud. Tenant separation is enforced logically (domains), and all API calls check whether the caller has access to the object. The ability to integrate LDAP and SAML allows organizations to use existing identity management solutions for logging in to CloudStack. Although

CloudStack does not have settings as granular as the OpenStack policy engine, its role model is sufficient for common scenarios (a regular user cannot affect another, a domain-admin sees only their own domain). A positive is the native 2FA support for the UI, which increases access security for administrative accounts [63]. Overall, CloudStack offers robust, albeit somewhat simpler, access management compared with OpenStack – this may be an advantage in terms of clarity. Security best practices (unique API keys for users, the possibility of key rotation) are implemented in CloudStack. It offers logging options, but auditing may be limited. Basic encryption is present.

Score: 4 pts

3.4.1.3 Proxmox VE

Description: Proxmox VE focuses primarily on virtualization management at the host level. It provides robust authentication mechanisms, including the internal Proxmox database system, LDAP/AD, PAM and Realm (for various authentication sources). Access control is based on granular RBAC, where roles and permissions can be defined at the level of VMs, storage, networking, etc. All administrative actions are logged and accessible via the GUI or CLI. Proxmox supports VM disk encryption (via LUKS) and secure communication (SSH, HTTPS for the web GUI). However, it is an infrastructure-oriented system, not one for cloud identity management for end users of services. It lacks advanced features such as self-service portals for users with multi-level authentication or automated security policies applied to the cloud service as a whole. Although Proxmox itself is very secure and regularly updated, for the needs of a comprehensive SOC solution with multi-tenant access it would be necessary to build additional IAM layers on top of it:

- **Multi-level authentication and authorization**: It supports multiple authentication sources, including PAM, LDAP, AD and OpenID Connect, with flexible access control
- **MFA support**: It can be integrated via external sources, but it is not natively supported.
- **Data encryption**: It supports encryption for VMs and containers, with integration of encrypted storage.
- **Logging and auditing**: It provides logging; auditing may require additional tools.
- **Automated security policies**: It allows permissions and ACLs to be defined for groups, with limited automation.
- **Modern security practices**: Regular updates; it supports modern standards such as OpenID Connect.
- **RBAC**: It implements RBAC via permissions and groups, enabling fine-grained control.

Justification: Proxmox is not primarily designed for a large number of separate tenants, but it contains all the important mechanisms for secure access management within a single organization. Integration with LDAP/AD allows corporate accounts to be used for login, which simplifies identity management. The granularity of RBAC in Proxmox is sufficient for smaller environments – the administrator can set very specific permissions (e.g. the right to reboot a specific VM without the ability to delete it). Moreover, 2FA support directly in the Proxmox interface significantly reduces the risk of unauthorized access, which is not a common standard in many similar tools. A limitation may be that Proxmox does not provide hard multi-tenancy isolation – if several different customers were to share the same Proxmox cluster, administrative accounts would have to be set up very carefully.

Score: 4 pts

3.4.1.4 Kubernetes

Description: Kubernetes has a very advanced system of authentication and access control. Authentication is flexible and can include certificates, tokens, LDAP/AD and OAuth 2.0 (via external integrations). Authorization is based on robust RBAC (Role-Based Access Control), which allows extremely granular definition of permissions at the level of the cluster, namespaces and individual resources (pods, deployments, services, etc.). All interactions with the Kubernetes API are logged in the audit log, which provides a detailed overview of activities. It supports encryption of the etcd database, encryption at rest for secrets and network segmentation (Network Policies). It is designed with an emphasis on security in a cloud-native environment and has an active community that continuously develops security tools (e.g. policy enforcement tools such as Gatekeeper and Kyverno). Although its focus is on containers, its security principles are highly relevant to data protection in a SOC:

- **Multi-level authentication and authorization**: It supports certificates, tokens, OIDC and service account authentication, with integration of external providers
- **MFA support**: It can be integrated via OIDC with providers that support MFA.
- **Data encryption**: It supports TLS for data in transit and encryption at rest for etcd.
- **Logging and auditing**: Built-in auditing and logging of all requests to the API server.
- **Automated security policies**: It uses RBAC and other mechanisms to enforce policies.
- **Modern security practices**: Regular security updates; it supports Pod Security Policies and Network Policies.
- **RBAC**: Fully implemented RBAC, enabling fine-grained access control.

Justification: It provides advanced and flexible multi-level authentication and extremely robust, granular access control (RBAC). It has full support for detailed logging and auditing of API events. It actively supports encryption and the implementation of automated security policies, which makes it an ideal choice for a cybersecurity environment

Score: 5 pts

3.4.2 K10: Tenant isolation and data protection (weight: 5)

Justification: In a multi-tenant environment, it is essential that the platform guarantees strict isolation between individual customers (tenants) – their virtual machines, storage and networks must be separated so as to prevent unauthorized access or data leakage between tenants. This criterion evaluates the platform's architecture in terms of multi-layered security: separation of compute resources, network segmentation, access control mechanisms and the like. The degree of data isolation between tenants is a fundamental aspect of multi-tenant systems.

Any weakness in isolation could lead to security incidents; we therefore assign this criterion **a weight of 5** (very high).

Scoring recommendation:

- **5 points:** High tenant isolation, complete resource segmentation, advanced data protection mechanisms.
- **3 points:** Partial isolation with some limitations, less robust protection mechanisms.
- **1 point:** Weak tenant isolation, risk of data leakage, weak access control.

3.4.2.1 OpenStack

Description: OpenStack was designed from the ground up for multi-tenant environments – each user (or project/tenant) is separated at all levels. Virtual machines of

different projects run on hypervisors in isolation by means of the hypervisor layer (KVM provides both memory and processor isolation of VMs). Network isolation is handled in the Neutron service: tenants have their own virtual networks (VLAN, VXLAN), and there is no communication between them unless the cloud admin explicitly allows it (e.g. via a router with rules) – so by default one tenant cannot "see" into another's network. Storage and volumes (Cinder) are always assigned to a single project, and access to them is authenticated via Keystone, so one tenant has no access to another's disk volumes. In addition, OpenStack supports data encryption – for example, Cinder allows volumes to be encrypted at block level (with keys stored in the Barbican service), and Glance can store QCOW2 images with encryption. To protect data in transit, OpenStack provides end-to-end encryption of API communication and the possibility of deploying VPN for tenants (VPN-as-a-service). With regard to a **SOCaaS (Security Operations Center as a Service)** deployment – which requires a multi-tenant and security-oriented environment – OpenStack has significant advantages. It accounts for multi-tenancy by design: tenant isolation is ensured at the level of virtual machines (full software isolation of VMs), virtual networks (private networks, VLAN/VXLAN segregation via Neutron) and storage (separate volumes and objects). Each tenant (project) has its own resources and identities managed via Keystone. **OpenStack is therefore capable of providing strong isolation and security mechanisms for multiple customers in a single cloud** [36]:

- **Compute isolation**: It uses hypervisors such as KVM, which ensure strong isolation of virtual machines (VMs) between tenants (projects). Each project has its own VMs, which are separated from the others (OpenStack Multi-Tenant Isolation).
- **Network segmentation**: The Neutron service supports isolated tenant networks via VLAN, VXLAN and security groups, thereby ensuring separation of network traffic (OpenStack Networking Security).
- **Storage isolation**: The Cinder (block storage) and Swift (object storage) services ensure separation of tenant data by means of project accounts and containers (OpenStack Security Guide).
- **Access control**: The Keystone service provides role-based access control (RBAC) and integration with LDAP/AD, thereby ensuring that users have access only to the resources of their own project.
- **Data protection**: It supports encryption of data in transit (TLS) and at rest (e.g. encrypted volumes in Cinder). The documentation describes procedures for hardening and data protection (OpenStack Hardening Guide).
- **Advanced mechanisms**: It includes security groups, firewall rules and integration with external security tools, thereby minimizing the risk of data leakage.
- **Known vulnerabilities**: OpenStack has processes for vulnerability management through OpenStack Security Advisories (OSSA) and the Vulnerability Management Team (VMT), which address security issues quickly (OpenStack Security Portal).

Justification: High tenant isolation is at the core of OpenStack. It provides complete segmentation of compute resources, networks and storage, together with advanced access control mechanisms (RBAC) and encryption, which minimizes the risk of data leakage between tenants.

Score: 5 pts

3.4.2.2 CloudStack

Description: CloudStack also provides multi-layered isolation between tenants (in CloudStack terminology, accounts or domains). At the network level, in "advanced" zones CloudStack

uses separate VLANs for each tenant's networks, or VXLAN for overlay networks – customers thus have a separate broadcast domain and address range. In “basic” zones (without VLANs), CloudStack instead separates VMs via security groups (L3 firewall rules similar to the AWS EC2 classic mode). Storage: disk volumes and templates are always associated with a specific account; for each account, CloudStack can restrict access to storage resources to its own VMs only. The admin can define **zones** and **domains**, which also allows customers to be segregated geographically or by infrastructure. Data protection is supported by functionalities such as disk encryption at the KVM hypervisor level (since version 4.11, support for LUKS encryption of volumes is available). CloudStack has a built-in system of **snapshots** and **backups**, which are also tied to the account – one tenant can neither see nor restore another's snapshot. High availability mechanisms (e.g. automatic VM restarts in the event of a host failure) contribute to the protection of data integrity by minimizing the risk of the entire system failing for a single client:

- **Compute isolation**: It uses hypervisors such as KVM and Xen, which ensure isolation of VMs between accounts or domains (CloudStack Networking).
- **Network segmentation**: It supports isolated networks and security groups, which ensure separation of tenants' network traffic by means of VLANs and virtual networks (CloudStack Security Groups).
- **Storage isolation**: It manages storage pools and volumes that are assigned to specific accounts, thereby ensuring data separation (CloudStack Administration).
- **Access control**: It implements RBAC via accounts and domains, with support for API keys, LDAP and SAML, thereby ensuring access control.
- **Data protection**: It supports TLS for data in transit and can integrate encrypted storage. The documentation describes security procedures (CloudStack Security).
- **Advanced mechanisms**: It offers network ACLs and security groups, although less advanced than OpenStack.
- **Known vulnerabilities**: Some CVEs: CloudStack has vulnerabilities, such as CVE-2024-42062, which have been fixed in newer versions (CloudStack Security Advisory).

Justification: In terms of isolation, CloudStack provides a degree of separation comparable to OpenStack in common scenarios. Each customer operates within its own domain, with separate networks and resources. CloudStack directly prevents the assignment of one public IP to two different accounts or the connection of a VM to a network to which it is not authorized. Network segmentation via VLAN/VXLAN is a reliable method also used in other clouds, guaranteeing that, for example, broadcast or ARP traffic does not leave the tenant. Data protection through encryption is not as comprehensively developed in CloudStack as in OpenStack (there is no equivalent of the Barbican service for centralized key management), but through integration with hypervisor tools CloudStack can also provide encrypted disks. Audits of deployments (Telia, BT and other telco operators use CloudStack) show that the platform is sufficiently secure for a multi-tenant environment. The only potential weakness may be the smaller ecosystem of security extensions – for example, fewer built-in options for IDS/IPS integration. Nevertheless, in the criterion of tenant isolation and data protection, CloudStack lags hardly at all behind OpenStack.

Score: 5 pts

3.4.2.3 Proxmox VE

Description: Proroxmox VE is not primarily designed for multiple tenants from different organizations, but rather for centralized virtualization within a single administrative domain. Isolation between VMs is ensured at the KVM hypervisor level – each VM runs in a separate QEMU process with its own isolated memory space. For LXC containers, Proxmox uses

isolation via Linux namespaces and cgroups; these provide separation, but share certain parts of the kernel with the host. As regards network isolation, Proxmox does not offer multi-tenant SDN out-of-the-box – however, the administrator can use VLAN tagging to separate the networks of different groups of VMs, or set up completely separate physical networks for different users. Data protection in a Proxmox environment relies mainly on backup and replication mechanisms: Proxmox enables scheduled VM backups (e.g. to Proxmox Backup Server) and VM replication between cluster nodes, which protects data against HW failure. VM disk encryption is not managed directly by Proxmox (it is left to the guest OS inside the VM, or to the storage level – ZFS encryption). When Ceph storage is used within a Proxmox cluster, native Ceph data protection mechanisms can be utilized:

- **Compute isolation**: Uses KVM and LXC, which ensure isolation of VMs and containers, but multi-tenancy requires additional configuration (Proxmox Multi-Tenancy).
- **Network segmentation**: Supports VLAN and OpenVSwitch for network isolation, but the configuration is manual and complex (Proxmox OVS Networking).
- **Storage isolation**: Storage can be assigned to specific VMs, but it is not natively designed for multi-tenancy.
- **Access control**: Supports RBAC and integration with LDAP/AD, but is not optimized for multi-tenant environments.
- **Data protection**: Supports encryption of VMs and backups, but advanced security features require external tools.
- **Advanced mechanisms**: Tools such as MultiPortal enable multi-tenancy, but they are not native (MultiPortal).
- **Known vulnerabilities**: CVEs such as SSRF have been fixed, but multi-tenant vulnerabilities are not specifically documented (Proxmox CVEs).

Justification: It has weak tenant isolation at the cloud level, because it is not primarily designed as a multi-tenant cloud platform. It provides VM isolation at the hypervisor level, but it lacks integrated mechanisms for automated segmentation and access control between different tenants without extensive external tools and manual effort. However, if Proxmox were to serve as a public cloud for unknown customers, the missing integrated barriers (e.g. one-click separate virtual networks for each tenant, automated firewall rules between tenants) represent a risk of misconfiguration and potential intrusion.

Rating: 1.5 pts

3.4.2.4 Kubernetes

Description: Kubernetes is focused on the isolation of containers and microservices, which is critical for a multi-tenant environment, although not primarily for IaaS virtual machines. It provides strong isolation mechanisms through Namespaces (logical separation of resources), Network Policies (granular network segmentation between pods/namespaces), Resource Quotas (limiting resource consumption), Security Contexts and Pod Security Standards (container security). RBAC (Role-Based Access Control) is extremely robust and enables very fine-grained control of access to the API as well as to resources within the cluster. All interactions are audited. Although VM isolation is possible through KubeVirt, Kubernetes isolation natively concerns containers. However, the isolation principles and tools that Kubernetes provides are highly portable and also applicable to the security aspects of multi-tenancy if the solution is built on containers or VMs under Kubernetes:

- **Compute isolation**: Containers share the kernel, but namespaces and tools such as GKE Sandbox ensure isolation (Kubernetes Multi-Tenancy).

- **Network segmentation**: Network policies restrict traffic between namespaces, thereby ensuring isolation (GKE Multi-Tenancy).
- **Storage isolation**: Persistent Volumes and Claims are bound to namespaces, thereby ensuring data separation.
- **Access control**: RBAC and Pod Security Standards ensure fine-grained access control (Kubernetes Security).
- **Data protection**: Supports encryption of data in transit (TLS) and at rest (etcd), with advanced security policies.
- **Advanced mechanisms**: Pod Security Policies, admission controllers and tools such as GKE Sandbox increase security (GKE Sandbox).
- **Known vulnerabilities**: Vulnerabilities are addressed quickly, with an emphasis on correct configuration (Kubernetes Vulnerabilities).

Justification: Kubernetes provides a high level of resource isolation (through Namespaces, Network Policies, RBAC) and advanced data protection mechanisms for container environments, which are equally critical for multi-tenancy. However, kernel sharing between containers carries certain risks – although container escape vulnerabilities are rare, they do exist.

Score: 4.5 pts

3.4.3 K11: Network/monitoring integration (weight: 3)

Justification: Justification: Evaluates the degree of integration with monitoring tools and security monitoring systems – in particular SIEM, IDS/IPS (e.g. Suricata, Snort), as well as metrics/logging tools such as Ceilometer, Grafana, Prometheus. It also takes into account the platform's ability to export security events, support for APIs/webhooks and the options for integration with external security components. The criterion complements the evaluation of authentication and isolation with the platform's ability to be part of an active security environment.

Scoring recommendation:

- 5 points: Full integration with tools (SIEM, IDS/IPS, Grafana, Prometheus), simple configuration.
- 3 points: Partial integration, some important tools are supported to a limited extent.
- 1 point: Very limited integration, problems with connecting tools and exporting events.

3.4.3.1 OpenStack

Description: OpenStack provides extensive options for integration with monitoring and security tools. For performance and resource monitoring, it has the Ceilometer (Telemetry) component, which collects metrics from various services (VM CPU load, disk usage, network traffic) and can store them or forward them. Ceilometer is complemented by Aodh (alarms) – it allows alarms to be defined on certain metrics, e.g. to notify if a VM exceeds 90% CPU. In addition, there is the Monasca project, which is a complete monitoring solution (time-series database, Grafana dashboards) integrated into OpenStack. OpenStack services can publish events and logs that can be connected to a SIEM – e.g. Keystone generates audit events in CADF format, usable in security monitors. For network integration, OpenStack offers the Tap-as-a-Service (TaaS) module for Neutron, which can mirror network traffic from a specific port to an analyser (IDS system) within the cloud. Furthermore, it is possible to deploy virtual network functions (VNFs) such as a firewall or IDS (e.g. a Suricata VM) and connect them into the tenant network topology. The OpenStack API allows external tools (Zabbix, Prometheus) to periodically request resource status and statistics – many modern integrations use the

Prometheus OpenStack exporter directly, which makes OpenStack metrics available to the Prometheus/Grafa system:

- **SIEM integration**: There is a specific application, Splunk App for OpenStack Analytics, which enables event monitoring and analysis (OpenStack Analytics Tool — Crest Data). The ELK stack (Elasticsearch, Logstash, Kibana) is also well integrated, with detailed setup guides (OpenStack Monitoring With ELK).
- **IDS/IPS**: Supports deployment of IDS/IPS systems on virtual machines, with the option of using Tap-as-a-Service (TaaS) for port mirroring in Neutron (Mirantis Documentation: IDS/IPS).
- **Metrics/logging tools**: It has built-in Ceilometer for telemetry, and integrates with Prometheus and Grafana, with dashboards available (Monitoring an Openstack deployment with Prometheus and Grafana).
- **Event export and API**: Supports export of security events via API and webhooks, with robust documentatio

Justification: Full integration with monitoring tools (Ceilometer, Grafana, Prometheus) and security systems (SIEM, IDS/IPS). It provides an extensive API and mechanisms for exporting security events and flexible configuration, which is key for an active SOC security environment. However, the complexity of OpenStack itself may require some effort during the initial configuration of these integrations.

Score: 5 pts

3.4.3.2 CloudStack

Description: CloudStack has a built-in Event and Alert system that records various events (e.g. VM creation, host failure) and can notify administrators about them. Integration with external monitoring tools is supported via the **Event Notification Framework** – CloudStack can publish all events to an AMQP message bus (e.g. RabbitMQ) or Apache Kafka, from where third-party systems can consume them. In this way, CloudStack can be connected to a SIEM – every significant event (adding a user, stopping a VM, a hypervisor error) appears as a message on the bus, which the SIEM evaluates. For performance monitoring, CloudStack directly offers the option of SNMP traps – it has defined MIBs for host and VM statistics, which it can send to SNMP-compatible monitoring. Furthermore, there are community plugins/exporters, for example for Zabbix (official template) or Prometheus, which read metrics via the CloudStack API and make them available for visualization . For network integration, CloudStack offers the option of connecting physical devices: it can be configured to work with external firewalls and load balancers (e.g. Palo Alto, Netscaler) – these devices then provide advanced traffic monitoring. CloudStack also supports so-called **Traffic Mirroring** by defining a policy on a port profile (although this feature is not as well documented as OpenStack TaaS).

- **SIEM integration**: It has no direct native integrations with SIEM systems, but logs can be exported to SIEM systems via third parties such as ELK, with plugins such as Log View
- The documentation indicates limited support for advanced SIEM functions.
- **IDS/IPS**: Enables deployment of IDS/IPS systems on virtual machines, with network traffic configuration, but it is not as directly supported as in OpenStack.
- **Metrics/logging tools**: Supports monitoring via tools such as Zabbix and Zenoss, but integration with Prometheus and Grafana may require additional setup.
- **Event export and API**: It has an API for log export, but the configuration may be more complex.

Justification: Compared to OpenStack Telemetry, CloudStack does not have a native component collecting metric time series – it relies rather on external tools to which it provides

the necessary data. This is a slight minus if we expected fully integrated graphs and alerts directly in the platform (the CloudStack UI displays only basic usage statistics). As regards security integration, CloudStack does not offer specific modules for IDS, but it allows an IDS to be deployed as part of the virtual infrastructure and events/alerts to be used for response – for example, an alert about a CPU threshold being exceeded on a VM may be an indicator of an attack and could trigger a scaling script via a webhook. Overall, CloudStack meets the requirements for connection to monitoring decently, although not as comprehensively out-of-the-box as some competitors.

Rating: 3.5 pts

3.4.3.3 Proxmox VE

Description: Proxmox VE offers built-in monitoring of basic metrics and syslog for event logging, but it does not have a directly integrated connection to external monitoring systems. The administration web interface displays the current CPU, memory and IO utilization as well as network traffic for each node and VM – this data is stored in RRD databases and can be exported. For more advanced monitoring, external tools are used in practice: a Prometheus exporter (an open-source project) is available for Proxmox, which can collect data from the Proxmox API and make it available to Prometheus. Similarly, there are templates for Zabbix and integrations for Check_MK/Icinga, which many administrators deploy to monitor Proxmox clusters. As regards network integration, Proxmox itself does not contain virtual networks with advanced features, so integration with IDS/IPS consists of deploying such tools as a VM or container. However, it is possible to take advantage of the fact that Proxmox runs on Linux – the admin can use Linux tools on the host (tcpdump, the iptables mirror module) if traffic capture is needed. Logs from Proxmox (including audit logs – logins, user operations) can be redirected to an external syslog/SIEM server for centralized monitoring. In addition, Proxmox supports API hook scripts – on certain events (e.g. VM start/stop) it can run a custom script, which can be used to integrate a response in an external system:

- SIEM integration: It can be integrated with SIEM tools such as Wazuh, with detailed setup guides . Logs can be sent to SIEM systems via syslog.
- IDS/IPS: Enables IDS/IPS systems to be run as virtual machines, with port mirroring configuration, as described in the guides .
- Metrics/logging tools: It has built-in monitoring, and integrates with Grafana and InfluxDB, with dashboards available . Prometheus is also supported via exporters.
- Event export and API: Supports log export via API and syslog, but the configuration may require additional steps.

Justification: Compared to cloud platforms, Proxmox does not have sophisticated out-of-the-box monitoring or an event system, but its openness allows admins to build the necessary connections themselves. For example, deploying Prometheus+Grafana on top of a Proxmox cluster is fairly well documented by the community and provides a complete overview of resources (CPU utilization, VM uptime, etc.). For smaller installations this is often not even necessary – the built-in graphs in the UI are sufficient for routine monitoring. From the security monitoring perspective, Proxmox relies on standard mechanisms: logging of all actions and the option to centralize them. Proxmox indicates security incidents (e.g. live migration failure, node outage) by an alarm in the UI and an e-mail notification, but it does not offer integration with an IDS. This reflects Proxmox's target area – deployment in a trusted environment, where the main goal is simplicity, not comprehensive security monitoring. If needed, however, nothing prevents Proxmox from being connected to a monitoring system of the organization's choice. In summary, we rate the integration of Proxmox with monitoring as satisfactory, with added effort on the part of administrators (the use of external tools and exporters is necessary).

Rating: 3.5 pts

3.4.3.4 *Kubernetes*

Description: Kubernetes is focused on observability and has excellent integration options with tools for monitoring, logging and tracing. The standard is integration with Prometheus (for metrics), Grafana (for visualization) and the ELK stack (Elasticsearch, Logstash, Kibana) or Loki/Fluentd (for logs). Kubernetes Audit Logs provide detailed records of all interactions with the API server, which are key for SIEM systems. Integration with IDS/IPS is possible through tools such as Falco (for runtime security and threat detection in containers), Calico for network policies and other CNI plugins that enable advanced network functions and monitoring. Kubernetes has an extensive and consistent API that enables programmatic integration with virtually any external system through operators and webhooks. It is ideal for the deployment and integration of containerized SIEM/IDS/IPS agents:

- SIEM integration: It can be integrated with SIEM systems via logs and tools such as Wazuh, Falco, with detailed guides . Audit logs can be exported to SIEM systems.
- IDS/IPS: Supports deployment of IDS/IPS systems as pods, with tools such as Falco for anomaly detection.
- Metrics/logging tools: It has standard integration with Prometheus and Grafana, with many pre-built dashboards .
- Event export and API: Supports export of events via API and webhooks, with robust documentation.

Justification: Kubernetes is one of the most monitorable platforms of all – metrics are available even in the basic installation, along with a huge selection of integrations within the CNCF project (Prometheus, Grafana, Loki for logs, Jaeger for tracing). For operators, this means that monitoring of a Kubernetes cluster and the applications in it can be carried out in a uniform way, including VMs thanks to KubeVirt. Since 80%+ of organizations using Kubernetes also deploy cloud-native monitoring cncf.io, there is a wealth of knowledge and support for setting up the right metrics and alerts. From a security perspective, Kubernetes does not lag behind: projects such as Falco can effectively detect potential attacks in the runtime environment, and network policies can restrict an attacker's movement in the event of a compromise of a single pod. Although KubeVirt brings classic VMs into the Kubernetes environment, it does not complicate integration – for example, Falco also sees suspicious activities in KubeVirt pods (e.g. if there were a breakout from a VM into a container). In addition, the cloud-native approach facilitates automated response: events from Kubernetes (whether monitoring or security events) can be responded to using Kubernetes Controllers or functions (e.g. automatically restart a pod, scale, block an IP). In view of these arguments, we rate the monitoring and network monitoring integrations in Kubernetes + KubeVirt as excellent – both the community and the tools are ready to cover the needs of a production deployment.

Score: 5 pts

4 EVALUATION TABLE FOR THE SELECTION OF A CLOUD PLATFORM

The following table presents the results of the multi-criteria evaluation of four candidate cloud platforms (OpenStack, Apache CloudStack, Proxmox VE and Kubernetes with KubeVirt) according to the eleven selection criteria defined in document V2. Each criterion is assigned a weight expressing its importance (in the range 1–5).

The evaluation was carried out on the basis of an expert analysis of available sources, documentation (many sources recur), case studies and academic publications. For each platform, each criterion was given a score on a scale of 1 to 5 points, which was then multiplied by the weight of the given criterion.

The overall score is therefore the weighted sum of all ratings. The platform with the highest total of points represents the most suitable solution for the given type of deployment from the perspective of the selected criteria.

		Openstack		CloudStack		Proxmox VE		Kubernetes	
Kritérium (Kx)	**Váha**	**Hodnotenie**	**Body celkom**	**Hodnotenie**	**Body celkom**	**Hodnotenie**	**Body celkom**	**Hodnotenie**	**Body celkom**
K1: Podpora IaaS funkcionality	5	5	25	4,5	22,5	3	15	1	5
K2: Modularita a architektúra	4	4,5	18	3	12	1	4	5	20
K3: Orchestrácia a automatizácia	3	5	15	4	12	3	9	5	15
K4: Výkonnosť a škálovateľnosť	5	5	25	5	25	3	15	5	25
K5: Spoľahlivosť a stabilita	4	5	20	5	20	4,5	18	5	20
K6: Štandardy a interoperabilita	4	5	20	3,5	14	3	12	4,5	18
K7: Náročnosť nasadenia a správy	3	1	3	4	12	5	15	1	3
K8: Dokumentácia a komunita	4	5	20	3,5	14	4	16	5	20
K9: Autentifikácia a prístupy	5	5	25	4	20	4	20	5	25
K10: Izolácia tenantov a ochrana dát	5	5	25	5	25	1,5	7,5	4,5	22,5
K11: Sieťová / monitorovacia integrácia	3	4,5	13,5	3,5	10,5	3,5	10,5	5	15
Súčet			209,5		187		142		188,5

The result of the analysis carried out of four candidate open-source platforms (OpenStack, Apache CloudStack, Proxmox VE and Kubernetes / KubeVirt) on the basis of 11 defined criteria is the identification of their strengths and weaknesses from the perspective of the requirements of the SOCaaS (Security Operations Center as a Service) project in a multi-tenant environment.

The platforms were evaluated in the areas of IaaS functionality, modularity, scalability, reliability, interoperability, security and operational demands. The evaluation followed a weighted approach, in which critical areas (e.g. security, performance, tenant isolation) were assigned a higher weight in line with the project objectives.

OpenStack achieved the highest weighted score in most categories. It is a fully fledged IaaS platform with a mature ecosystem of components (Compute, Networking, Storage), which relies on a large open-source community, support for standards and experience from large-scale production deployments. From a security perspective, it provides precise mechanisms for authentication, authorization, isolation and encryption. Although its deployment is technically demanding, the return on investment in a robust, modular and scalable solution outweighs the initial costs in an environment with long-term service development.

Apache CloudStack offers lower deployment complexity, very good reliability and proven operational stability. It represents a reasonably balanced alternative with lower complexity and more accessible administration, while retaining the ability to provide basic IaaS functions and security features suitable for a multi-tenant environment. However, it lags behind in terms of flexibility, extensibility and strong support from the wider community. It came second in the comparison.

Proxmox VE is an easily deployable virtualization solution, ideal for teaching, testing or internal infrastructures. Its advantages lie in fast installation, low administration demands and stable performance for smaller environments. However, Proxmox is not designed for large-scale multi-tenant cloud operations with multiple external customers requiring rigorous tenant isolation, SLA support, advanced orchestration or hybrid service development.

Kubernetes brings the concept of containerized and virtualized workloads under a single orchestrator. It is exceptionally strong in the areas of scaling, automation and integration of monitoring tools. Despite its growing maturity, however, it runs into certain limits in the context of the project: VM support is secondary, traditional IaaS mechanisms are missing (e.g.

a comprehensive storage API, separate tenant management, native network virtualization at the VM level) and the security model is not primarily designed for hostile multi-tenancy without strong security built around it. It was included in the comparison as a representative of the container approach.

5 FINAL SUMMARY

On the basis of the multi-criteria analysis carried out and the assessment of the results of the individual platforms, we recommend deploying OpenStack as the main technical platform for the implementation of the target SOCaaS solution. OpenStack received the highest rating reflecting its level of technical maturity, flexibility, security and compatibility with standards. It is capable of providing a robust IaaS infrastructure that enables the efficient building and operation of a scalable, multi-tenant environment with an emphasis on security operations. The solution is suitable for the academic and research sphere, where long-term sustainability, openness, vendor independence and the possibility of involving community and research activities are important. As an alternative, in the event that operational conditions (staffing, time or infrastructure) do not allow the immediate deployment of OpenStack, we recommend Apache CloudStack. It represents a reasonably simplified compromise between functionality and operational simplicity, while retaining an acceptable level of security and tenant separation. We recommend deploying Proxmox VE only for limited internal or teaching scenarios without critical security or scaling requirements. Kubernetes with KubeVirt represents a promising extension in the event of a transition to a cloud-native model of SOC services, but it is not suitable as the primary IaaS solution for the current objective of the project.

LIST OF SOURCES

1] „OpenStack,“ [Online]. Available: https://www.openstack.org/.

2] „Apache CloudStack,“ [Online]. Available: https://cloudstack.apache.org/.

3] A. Vogel, D. Greibler and C. Maron, „Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack,“ rev. *Conference: 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 2016.

4] „CloudStack vs OpenNebula comparison,“ [Online]. Available: https://www.peerspot.com/products/comparisons/cloudstack_vs_opennebula.

5] „Proxmox VE,“ [Online]. Available: https://www.proxmox.com/en/proxmox-ve.

6] „Kubernetes,“ [Online]. Available: https://kubernetes.io/.

7] „Open Nebula,“ [Online]. Available: https://opennebula.io/.

8] „oVirt,“ [Online]. Available: https://www.ovirt.org/.

9] „Eucalyptus,“ [Online]. Available: https://www.eucalyptus.cloud/.

10] „Harvester,“ [Online]. Available: https://harvesterhci.io/.

11] „Kubevirt,“ [Online]. Available: https://kubevirt.io/.

12] „The Most Widely Deployed Open Source Cloud Software in the World,“ [Online]. Available: https://www.openstack.org/software/project-navigator/openstack-components#openstack-services.

13] A. Rabcan, *Návrh architektúry hybrid-cloudového prostredia pre,* Žilina: UNIZA, 2020.

14] S. Hopko, *Problematika Cloud computing a jeho využitia,* UNIZA, 2015.

15] „OpenStack Architecture Guide,“ [Online]. Available: https://docs.openstack.org/arch-design/.

16] I. Voras, B. Mihaljevic and M. Orlic, „Criteria for Evaluation of Open Source Cloud Computing Solutions,“ rev. *33rd International Conference on Information Technology Interfaces*.

17] „CloudStack vs OpenStack: Comparing Open Source Cloud Infrastructure Management Solutions,“ [Online]. Available: https://openmetal.io/resources/blog/cloudstack-vs-openstack/#:~:text=1.%20All,CloudStack.

18] „loudStack vs. OpenStack Comparison – What you need to know before choosing a cloud management system,“ [Online]. Available: https://www.shapeblue.com/cloudstack-vs-openstack-

comparison/#:~:text=Image%3A%20OpenStack%20ArchitectureOpenStack%20is%20broken,components%20depending%20on%20your%20needs.

[19] „Apache CloudStack. (n.d.). Architecture Overview,“ [Online]. Available: https://cloudstack.apache.org/.

[20] „Proxmox VE. (n.d.). Proxmox VE Architecture,“ [Online]. Available: https://pve.proxmox.com/wiki/Main_Page.

[21] „Proxmox vs OpenStack: Comparing Open Source Cloud Software,“ [Online]. Available: https://openmetal.io/resources/blog/proxmox-vs-openstack-comparing-open-source-cloud-software/#:~:text=1,provides%20a%20comprehensive%20suite%20of.

[22] „ubernetes. (n.d.). Kubernetes Architecture.,“ [Online]. Available: https://kubernetes.io/docs/concepts/architecture/.

[23] „OpenStack. (n.d.). Heat Orchestration Service.,“ [Online]. Available: https://docs.openstack.org/heat/.

[24] „Terraform Registry. (n.d.). OpenStack Provider.,“ [Online]. Available: https://registry.terraform.io/providers/OpenStack/openstack/latest/docs.

[25] „Ansible. (n.d.). OpenStack Modules,“ [Online]. Available: https://docs.ansible.com/ansible/latest/collections/openstack/cloud.

[26] „ Apache CloudStack. (n.d.). API Documentation.,“ [Online]. Available: https://cloudstack.apache.org/api/index.html.

[27] „Terraform Registry. (n.d.). CloudStack Provider.,“ [Online]. Available: https://registry.terraform.io/providers/cloudstack/cloudstack/latest/docs.

[28] „Ansible. (n.d.). CloudStack Modules.,“ [Online]. Available: https://docs.ansible.com/ansible/latest/collections/community/general/.

[29] „ Proxmox VE. (n.d.). Proxmox VE API Documentation,“ [Online]. Available: https://pve.proxmox.com/pve-docs/api-viewer/.

[30] „GitHub. (n.d.). Telmate/terraform-provider-proxmox,“ [Online]. Available: https://github.com/Telmate/terraform-provider-proxmox.

[31] „Ansible Galaxy. (n.d.). Proxmox modules/roles,“ [Online]. Available: https://galaxy.ansible.com/search?keywords=proxmox.

[32] „Kubernetes. (n.d.). Concepts: Workloads,“ [Online]. Available: https://kubernetes.io/docs/concepts/workloads/.

[33] „Kubernetes. (n.d.). Operators,“ [Online]. Available: https://kubernetes.io/docs/concepts/extend-kubernetes/operator/.

[34] „erraform Registry. (n.d.). Kubernetes Provide,“ [Online]. Available: https://registry.terraform.io/providers/hashicorp/kubernetes/latest/docs.

[35] „ Ansible. (n.d.). Kubernetes Modules. D,“ [Online]. Available: https://docs.ansible.com/ansible/latest/collections/kubernetes/core/.

[36] „1000 Compute nodes resource scalability testing,“ [Online]. Available: https://docs.openstack.org/developer/performance-docs/test_results/1000_nodes/index.html#:~:text=We%20have%208%20instances%20of,13%20nodes.

[37] „OpenStack. (n.d.). OpenStack Operations Guide - Scaling your Cloud,“ [Online]. Available: https://docs.openstack.org.

[38] „Capacity planning and scaling,“ [Online]. Available: https://docs.openstack.org/operations-guide/ops-capacity-planning-scaling.html.

[39] „Apache CloudStack. (n.d.). Architectural Overview,“ [Online]. Available: https://cloudstack.apache.org/.

[40] „Performance Evaluation of the CloudStack Private Cloud,“ rev. *DOI: 10.11591/closer.v2i6.5661*.

[41] „Proxmox VE. (n.d.). Proxmox VE Cluster,“ [Online]. Available: https://pve.proxmox.com/wiki/Cluster_Manager.

[42] „Clustering in Proxmox VE,“ [Online]. Available: https://hostman.com/tutorials/clustering-in-proxmox-ve/.

[43] „Proxmox forum,“ https://forum.proxmox.com/threads/proxmox-scalability-max-clusters-in-a-datacenter.122541/.

[44] „Kubernetes. (n.d.). Kubernetes Scale Targets,“ [Online]. Available: https://kubernetes.io/docs/setup/best-practices/cluster-large/.

[45] „Kubernetes. (n.d.). Autoscalin,“ [Online]. Available: https://kubernetes.io/docs/concepts/workloads/controllers/autoscaler/.

[46] „OpenStack Foundation. (n.d.). User Stories & Case Studies.,“ [Online]. Available: https://www.openstack.org/users/.

[47] „VMware vs OpenStack: Why OpenStack is the better VMware alternative in 2024,“ [Online]. Available: https://platform9.com/blog/openstack-vs-vmware/.

[48] „OpenStack. (n.d.). OpenStack Development Cycle and Testing,“ [Online]. Available: https://docs.openstack.org/dev-docs/.

[49] „Apache CloudStack. (n.d.). About CloudStack,“ [Online]. Available: https://cloudstack.apache.org/about.html.

[50] „Apache CloudStack. (n.d.). CloudStack Installation Guide - High Availability.,“ [Online]. Available: https://docs.cloudstack.apache.org/en/latest/installguide/ .

[51] „Apache CloudStack. (n.d.). User Stories.,“ [Online]. Available: https://cloudstack.apache.org/users/index.html.

[52] „Proxmox VE. (n.d.). About Proxmox VE,“ [Online]. Available: https://pve.proxmox.com/wiki/Main_Page.

[53] „Proxmox VE. (n.d.). High Availability Cluster,“ [Online]. Available: https://pve.proxmox.com/wiki/High_Availability_Cluster.

[54] „ubernetes. (n.d.). User Stories,“ [Online]. Available: https://kubernetes.io/case-studies/.

[55] „Kubernetes. (n.d.). High Availability Kubernetes Clusters,“ [Online]. Available: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/.

[56] „oudFerro builds its clouds based on OpenStack - widely deployed cloud framework,“ [Online]. Available: https://cloudferro.com/cloud/openstack/#:~:text=Industry,APIs.

[57] „CloudStack Installation options,“ [Online]. Available: https://docs.cloudstack.apache.org/en/4.17.0.0/installguide/optional_installation.html#:~:text=AWS%20API%20User%20Setup%C2%B6.

[58] „CNCF Research Reveals How Cloud Native Technology is Reshaping Global Business and Innovation,“ [Online]. Available: https://www.cncf.io/announcements/2025/04/01/cncf-research-reveals-how-cloud-native-technology-is-reshaping-global-business-and-innovation/#:~:text=CNCF%20Research%20Reveals%20How%20Cloud,CI%2FCD%20adoption%20is%20fueling.

[59] „https://openmetal.io/resources/blog/cloudstack-vs-openstack/#:~:text=1.%20All,set%20up%20and%20deploy%20compared,“ [Online].

Available: https://openmetal.io/resources/blog/cloudstack-vs-openstack/#:~:text=1.%20All,set%20up%20and%20deploy%20compared.

60] „CloudStack vs OpenStack: How Do They Compare To Each Othe,“ [Online]. Available: https://openmetal.io/resources/blog/cloudstack-vs-openstack/#:~:text=However%2C%20this%20flexibility%20comes%20at,times%20and%20increased%20operational%20complexity.

61] „OpenStack Keystone Authentication Using LDAP,“ [Online]. Available: https://platform9.com/blog/keystone-authentication-ldap/.

62] „OpenStack MFA,“ [Online]. Available: https://docs.openstack.org/keystone/latest/user/multi-factor-authentication.html.

63] „Apache CloudStack Features,“ [Online]. Available: https://cloudstack.apache.org/features/#:~:text=Multi.

64] „Kubernetes vs OpenStack: which one to choose?,“ [Online]. Available: https://ubuntu.com/blog/kubernetes-vs-openstack#:~:text=OpenStack%20is%20a%20cloud%20platform,institutions%2C%20manufacturing%20companies%20and%20governments.