PROJECT

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

Analysis of available open-source solutions for Infrastructure as a Service (IaaS) and container orchestration systems

Linked to work package KPB1 – Design of the virt. solution

Linked to output V1 – **Result of the analysis of available solutions**

CONTENTS

Contents 2
1. Introduction 3
2. Infrastructure as a Service (IaaS) platforms 4
2.1. Apache CloudStack 4
2.2. OpenStack 8
2.3. OpenNebula 11
2.4. Eucalyptus 14
2.5. Comparison of IaaS platforms 18
2.6. Evaluation 20
3. Container orchestration systems 21
3.1. Kubernetes (K8s) 21
3.2. Docker Swarm 24
3.3. Apache Mesos 26
3.4. HashiCorp Nomad 30
3.5. Comparison of container orchestration tools 33
3.6. Evaluation 35

1. INTRODUCTION

There are several open-source cloud computing platforms on the market that can be used for IaaS-type services ("Infrastructure as a Service") and container orchestration systems as an alternative to the commercial products of Google, IBM, Microsoft, Amazon, Alibaba and others. The following text provides a more detailed description and comparison of the most significant and most frequently used open-source solutions from among a wider range of solutions (Apache CloudStack, OpenStack, OpenNebula, Eucalyptus, Kuberntes, Docker Swarm, Apache Mesos, HaschiCorp Nomad, Red Hat OpenShift, oVirt, OpenQRM, Ceph, Nimbus and others.)

2. INFRASTRUCTURE AS A SERVICE (IAAS) PLATFORMS

2.1. APACHE CLOUDSTACK

Apache CloudStack is a powerful, feature-rich infrastructure as a service (IaaS) platform for building and managing cloud infrastructure, designed for the scalable and efficient deployment and management of large networks of virtual machines (VMs). Thanks to its flexibility, scalability and extensive feature set, it is suitable for a wide range of use cases, from small private clouds to large-scale public cloud deployments. A detailed summary of its key features is given below:

Core functions and components

1. Multi-tenancy

- **Purpose**: Supports multiple tenants (users, departments or organizations) with isolated resources and environments.
- **Key features**:
  - **Resource isolation**: Ensures that each tenant's resources are isolated from the others.
  - **Quotas**: Setting resource limits for individual tenants.
  - **Accounts and domains**: Organizes users into hierarchical domains and accounts for better resource allocation and management.

2. User management

- **Purpose**: Provides tools for managing users and their access to resources.
- **Key features**:
  - **Role-Based Access Control (RBAC)**: Define roles and permissions for users.
  - **LDAP/AD integration**: Works with LDAP and Active Directory for user authentication.
  - **Self-service portal**: Allows users to provision and manage their own resources.

3. Resource management

- **Purpose**: Efficiently manages compute, storage and network resources.
- **Key features**:
  - **Dynamic resource allocation**: Automatically allocates and scales resources based on demand.
  - **Resource pooling**: Groups physical resources into clusters, pods and zones for efficient management.
  - **Load balancing**: Distributes the load across multiple hosts.

4. Hypervisor support

- **Purpose**: Supports various hypervisors for managing virtual machines (VMs).
- **Key features**:
  - **Support for multiple hypervisors**: Bare-Metal (via IPMI), VMWare (via vCenter), KVM, XenServer, XenProject, Hyper-V and LXC.

- **Mixed hypervisor environments**: Allows different hypervisors to be used within the same cloud infrastructure.

5. Networking

- **Purpose**: Manages network connectivity and IP addresses for cloud resources.
- **Key features**:
  - **Advanced networking models**:
    - **Basic networking**: Simple, flat network configurations.
    - **Advanced networking**: Supports complex network topologies with VLANs, VPNs and private IP addresses.
  - **Network services**:
    - Load balancing (HAProxy, F5, Netscaler).
    - Firewall management (ingress and egress data rules).
    - VPN (site-to-site and remote access).
    - NAT (network address translation) and port forwarding.
  - **Software-defined networking (SDN)**: Integrates with SDN solutions such as Nuage Networks and Midokura.

6. Storage management

- **Purpose**: Manages storage resources for virtual machines and the cloud infrastructure.
- **Key features**:
  - **Storage types**:
    - **Primary storage**: For VM disks (local storage, NFS, iSCSI, Fiber Channel, etc.).
    - **Secondary storage**: For templates, ISO images and snapshots.
  - **Storage features**:
    - Snapshots: Point-in-time backups of virtual machines.
    - Templates: Preconfigured virtual machine images for rapid deployment.
    - Volume management: Allows volumes to be resized and attached to/detached from virtual machines.
  - **Storage plugins**: Supports integration with external storage systems such as Ceph, SolidFire and NetApp.

7. API and extensibility

- **Purpose**: Provides tools for automation and integration with other systems.
- **Key features**:
  - **RESTful API**: A comprehensive API for automating cloud operations and integrating with third-party tools.
  - **Customizability**: The ability to extend functionality by means of plugins and custom scripts.

- **CLI tools**: Command-line interface for advanced users and automation.

8. High availability and scalability

- **Purpose**: Ensures service continuity and scalability.
- **Key features**:
  - **High Availability (HA)**:
    - Virtual machine failover: In the event of a failure, automatically restarts virtual machines on another host.
    - Redundant management servers: Ensures continuous operation even if a management server fails.
  - **Scalability**:
    - Supports on the order of thousands of physical servers and tens of thousands of virtual machines.
    - Horizontally scalable architecture.

9. User interface

- **Purpose**: Provides a web-based user interface for managing cloud resources.
- **Key features**:
  - **Web-based UI**: An intuitive interface for managing and monitoring cloud resources.
  - **Dashboard**: Provides an overview of resource utilization, virtual machine status and performance metrics in real time.
  - **Self-service portal**: Allows end users to provision and manage their own virtual machines and resources.

10. Templates and ISO

- **Purpose**: Manages virtual machine templates and ISO images for rapid deployment.
- **Key features**:
  - **Virtual machine templates**: Preconfigured operating system images for rapid deployment of virtual machines.
  - **ISO support**: Allows the user to upload and boot from ISO images for custom operating system installations.
  - **Template sharing**: Templates can be shared between zones and accounts.

11. Billing and accounting

- **Purpose**: Tracks resource utilization for billing and accounting purposes.
- **Key features**:
  - **Usage metrics**: Tracks resource utilization (processor, memory, storage, network) for billing purposes.
  - **Integration**: Works with third-party billing systems such as jBilling and Cloudian.

12. Disaster recovery

- **Purpose**: Ensures data integrity and availability in the event of a failure.

- **Key features**:
  - **Backup and recovery**: Supports virtual machine snapshots, backups and recovery plans.
  - **Cross-zone replication**: Replicates data between zones for disaster recovery.

13. Security

- **Purpose**: Ensures secure access to and operation of cloud resources.
- **Key features**:
  - **Network isolation**: VLANs and private networks for secure communication.
  - **Encryption**: Supports SSL/TLS for secure access to the API and the user interface.
  - **Access control**: Fine-grained permissions for users and resources.
  - **Security groups**: Firewall rules for virtual machines to control incoming and outgoing traffic.

14. Multi-zone and multi-region support

- **Purpose**: Manages resources across multiple geographical areas.
- **Key features**:
  - **Regions**: A collection of one or more geographically proximate zones managed by one or more management servers.
  - **Zones:** A zone is usually equivalent to a single data centre. A zone consists of one or more pods and secondary storage.
  - **Pods:** A pod is usually a rack or a group of racks containing a layer 2 switch and one or more clusters.
  - **Cross-zone workloads**: Allows the load to be distributed across zones to ensure redundancy and performance.

15. Ecosystem integration

- **Purpose**: Integrates with existing IT infrastructure and tools.
- **Key features**:
  - **Orchestration tools**: Integrates with tools such as Kubernetes, Terraform and Ansible.
  - **Monitoring**: Works with monitoring solutions such as Grafana, Prometheus and Nagios.
  - **CloudStack Ecosystem**: Plugins and extensions for additional functionalities.

Advantages

- **Flexibility**: Supports multiple hypervisors, storage and networking options.
- **Scalability**: Designed for large-scale deployments.
- **Ease of use**: Intuitive user interface and comprehensive API for management and automation.

2.2. OPENSTACK

OpenStack is an open-source cloud computing platform that provides a set of software tools for building and managing public, private and hybrid clouds. It is designed to control large pools of compute, storage and networking resources in a data centre, all managed through web-based visualization tools (dashboards), command-line tools or RESTful APIs. OpenStack is highly modular, with each component serving a specific function, and it is widely used by enterprises, service providers and developers to build scalable and flexible cloud environments.

Core functions and components

OpenStack consists of several interconnected projects, each of which is responsible for a specific aspect of the cloud infrastructure (Nova, Swift, Horizon, ...). A detailed breakdown of its core components is given below:

1. Compute (Nova)

- **Purpose**: Manages the life cycle of virtual machines (VMs) and provides compute resources; it is used for provisioning, scaling and managing virtual machines.
- **Key features**:
    - **Support for multiple hypervisors**: KVM, XLC, QEMU, UML, VMWare, Hyper-V, Xen, XenServer, Virtuozzo
    - **Live Migration**: Moves running virtual machines between physical hosts without downtime.
    - **Scalability**: Horizontally scalable to support thousands of virtual machines.
    - **Resource scheduling**: Allocates compute resources based on availability and defined policies.

2. Data storage

- **Object storage (Swift)**:
    - **Purpose**: Provides scalable and redundant storage for unstructured data (e.g. images, backups).
    - **Key features**:
        - **Resilience**: Replicates data across multiple nodes for high availability.
        - **Scalability**: Can scale up to petabytes of data.
        - **RESTful API**: Enables programmatic access to stored objects.
- **Block storage (Cinder)**:
    - **Purpose**: Provides persistent block storage volumes for virtual machines, database storage, virtual machine boot volumes and data-intensive applications.
    - **Key features**:
        - **Volume management**: Creating, attaching and managing storage volumes.
        - **Snapshots**: Point-in-time backups of volumes.
        - **Multiple backends**: Supports Ceph, NetApp, SolidFire and others.

- **Shared file systems (Manila)**:
    - **Purpose**: Offers shared file storage for virtual machines and applications.
    - **Key features**:
        - **Protocol support**: NFS, CIFS and others.
        - **Multi-tenancy**: Isolates file systems for different users or projects.

3. Network management (Neutron)

- **Purpose**: Manages network connectivity and IP addresses for cloud resources, network isolation, secure communication and traffic management.
- **Key features**:
    - **Software-defined networking (SDN)**: Abstraction of network configuration and management.
    - **Advanced networking**:
        - VLANs, VPNs and private networks.
        - Load balancing, firewalls and security groups.
    - **Plug-ins**: Integrates with SDN solutions such as Open vSwitch, Cisco ACI and Juniper Contrail.
- **Use cases**:

4. Identity and access management (Keystone)

- **Purpose**: Provides authentication, authorization and service discovery.
- **Key features**:
    - **Authentication**: Supports multiple methods (e.g. passwords, tokens, OAuth).
    - **Role-based access control (RBAC)**: Defines user roles and permissions.
    - **Multi-tenancy**: Isolates resources for different projects and users.
    - **Integration**: Works with LDAP, Active Directory and other identity providers.
- **Use cases**: User management, secure access control and multi-tenant environments.

5. Dashboard (Horizon)

- **Purpose**: Provides a web-based user interface for managing cloud resources.
- **Key features**:
    - **Self-service portal**: Enables users to provision and manage resources.
    - **Customizable**: Administrators can customize the dashboard to specific needs.
    - **Real-time monitoring**: Displays resource utilization metrics and performance metrics.
- **Use cases**: Simplified cloud management and monitoring.

6. Image management service (Glance)

- **Purpose**: Stores and manages disk images for virtual machines.
- **Key features**:

  - **Template support**: Preconfigured virtual machine images for rapid deployment.
  - **Backend support**: Works with Swift, Ceph and other storage backends.
  - **Versioning**: Tracks multiple versions of disk images.
- **Use cases**: Virtual machine provisioning, image sharing and template management.

7. Orchestration (Heat)

- **Purpose**: Automates the deployment and management of cloud resources.
- **Key features**:
  - **Infrastructure as code**: Uses templates to define and deploy resources.
  - **Compatibility with AWS CloudFormation**: Supports AWS templates.
  - **Auto-scaling**: Automatically scales resources based on demand.
- **Use cases**: Automated deployments, scaling and resource management.

8. Telemetry (Ceilometer)

- **Purpose**: Collects and monitors resource usage data.
- **Key features**:
  - **Metering**: Tracks usage for billing and chargeback purposes.
  - **Alarms**: Triggers actions based on predefined thresholds.
  - **Integration**: Works with monitoring tools such as Grafana and Prometheus.
- **Use cases**: Billing, monitoring and performance optimization.

9. Database service (Trove)

- **Purpose**: Provides relational and non-relational database instances.
- **Key features**:
  - **Supported databases**: MySQL, PostgreSQL, MongoDB, Cassandra and others.
  - **Automated management**: Handles backup, restore and scaling.
- **Use cases**: Database as a service (DBaaS) for applications.

10. Bare-Metal support (Ironic)

- **Purpose**: Manages Bare-Metal servers ("bare iron" servers without an OS) as if they were virtual machines.
- **Key features**:
  - **Physical server management**: Provisions and manages bare-metal servers.
  - **Use cases**: High-performance workloads, container orchestration (e.g. Kubernetes).

11. Containers (Magnum)

- **Purpose**: Simplifies the deployment of container orchestration tools.
- **Key features**:
  - **Supported tools**: Kubernetes, Docker Swarm and Mesos.

- **Cluster management**: Automates the deployment of container clusters.
- **Use cases**: Containerized applications and microservices.

12. DNS service (Designate)

- **Purpose**: Provides DNS as a service.
- **Key features**:
  - **Zone management**: Manages DNS zones and records.
  - **Integration**: Works with backend DNS servers such as BIND and PowerDNS.
- **Use cases**: DNS management for cloud resources.

13. Messaging service (Zaqar)

- **Purpose**: Provides a multi-tenant messaging service.
- **Key features**:
  - **Message queuing**: Supports event-driven architectures.
  - **Use cases**: Microservices, notifications and workflows.

14. Workflow service (Mistral)

- **Purpose**: Defines and executes workflows for complex tasks.
- **Key features**:
  - **Task automation**: Automates repetitive tasks.
  - **Integration**: Works with other OpenStack services and external systems.
- **Use cases**: Orchestration, automation and task management.

Additional features

- **Multi-region support**: Manages resources across multiple geographical regions.
- **High availability**: Ensures service continuity with redundant components and failover mechanisms.
- **Security**: Implements encryption, secure APIs and network isolation.
- **Extensibility**: Supports plug-ins and integrations with third-party tools and services.

Advantages

- **Flexibility**: Supports a wide range of hypervisors, storage and networking options.
- **Scalability**: Designed for large-scale deployments.
- **Ecosystem**: Integrates with a wide range of tools and technologies.

2.3. OPENNEBULA

OpenNebula is a powerful and flexible cloud computing platform known for its simplicity, flexibility and ability to integrate with existing IT infrastructure. OpenNebula is widely used by enterprises, service providers and research institutions to build private, public and hybrid clouds. It provides a unified interface for managing virtualized resources, including compute (computing), storage and networks. OpenNebula consists of several interconnected components, each of which is responsible for a specific aspect of the cloud infrastructure.

Core functions and components

1. Virtual infrastructure management

- **Purpose**: Manages the life cycle of virtual machines (VMs) and provides compute resources.
- **Key features**:
  - **Support for multiple hypervisors**: Works with KVM, VMware, Xen and LXD.
  - **Virtual machine life cycle management**: Creating, deploying, cloning, migrating and terminating virtual machines.
  - **Live Migration**: Enables running VMs to be moved between physical hosts without service downtime.
  - **Snapshots**: Enables "snapshots" of virtual machines to be created and managed for backup and recovery.
  - **Resource scheduling**: Allocates compute resources based on availability and policies.
- **Use cases**: Provisioning, scaling and managing virtual machines.

2. Storage management

- **Purpose**: Manages storage resources for virtual machines and the cloud infrastructure.
- **Key features**:
  - **Multiple storage backends**: Supports local, NFS, iSCSI, Ceph and others.
  - **Image management**: Centralized repository for virtual machine images and templates.
  - **Disk management**: Enables disks to be attached, detached and resized for virtual machines.
  - **Storage snapshots**: Creating snapshots of virtual machine disks for backup.
  - **Storage quotas**: Enables storage limits to be set for users and groups.
- **Use cases**: Virtual machine storage, image repositories and data backup.

3. Network management

- **Purpose**: Manages network connectivity and IP addresses for cloud resources.
- **Key features**:
  - **Virtual networks**: Creating and managing isolated virtual networks.
  - **Network isolation**: VLANs and private networks for secure communication.
  - **Advanced network management**: Supports SDN (software-defined networking) and integrates with Open vSwitch.
  - **Security groups**: Enables firewall rules to be defined for virtual machines.
  - **Network quotas**: Settings of network resource limits for users and groups.
- **Use cases**: Network isolation, secure communication and traffic management.

4. User and group management

- **Purpose**: Provides authentication, authorization and resource isolation.
- **Key features**:
  - **Multi-tenancy**: Supports multiple users and groups with isolated resources.
  - **Role-based access control (RBAC)**: Define roles and permissions for users.
  - **Quotas**: Set resource limits for users and groups.
  - **LDAP/AD integration**: Works with LDAP and Active Directory for user authentication.
- **Use cases**: User management, secure access control and multi-tenant environments.

5. Self-service portal

- **Purpose**: Provides a web-based user interface for managing cloud resources.
- **Key features**:
  - **Self-service provisioning**: Users can provision and manage their own virtual machines.
  - **Customizable**: Enables the portal to be customized to suit specific needs.
  - **Real-time monitoring**: Displays resource utilization and performance metrics.
  - **Template-based provisioning**: Enables the use of predefined templates for rapid deployment of virtual machines.
- **Use cases**: Simplified cloud management and monitoring.

6. Automation and orchestration

- **Purpose**: Automates the deployment and management of cloud resources.
- **Key features**:
  - **Infrastructure as code**: Enables infrastructure to be defined and deployed using templates.
  - **Workflow automation**: Helps automate repetitive tasks and workflows.
  - **Integration**: Works with tools such as Ansible, Terraform and Kubernetes.
  - **Cloud Bursting**: Integration with public clouds such as AWS and Azure for hybrid cloud deployment.
- **Use cases**: Automated deployments, scaling and resource management.

7. Monitoring and accounting

- **Purpose**: Collects and monitors resource usage data.
- **Key features**:
  - **Resource monitoring**: Tracking of processor, memory, disk and network utilization.
  - **Usage reporting**: Generating reports for billing and chargeback.
  - **Alerts**: Enables alerts to be set for resource utilization and performance metrics.
  - **Integration**: Can work with monitoring tools such as Nagios and Zabbix.
- **Use cases**: Billing, monitoring and performance optimization.

8. High availability and scalability

- **Purpose**: Ensures service continuity and scalability.
- **Key features**:
  - **High availability (HA)**: Redundant components and failover mechanisms.
  - **Scalability**: Designed to scale from small deployments to large data centres.
  - **Load balancing**: Distributed workloads across multiple hosts.
- **Use cases**: High availability, disaster recovery and scalable cloud infrastructure.

9. Security

- **Purpose**: Ensures secure access to and operation of cloud resources.
- **Key features**:
  - **Network isolation**: VLANs and private networks for secure communication.
  - **Encryption**: Supports SSL/TLS for secure access to the API and the user interface.
  - **Access control**: Fine-grained permissions for users and resources.
  - **Security groups**: Defines firewall rules for virtual machines.
- **Use cases**: Secure cloud environments and compliance.

10. Ecosystem integration

- **Purpose**: To integrate with existing IT infrastructure and tools.
- **Key features**:
  - **Cloud Bursting**: Integration with public clouds, such as AWS and Azure, for hybrid cloud deployment.
  - **Container support**: Enables containers to be managed alongside virtual machines.
  - **API and CLI**: Comprehensive API and command-line tools for automation and integration.
  - **Third-party tools**: Works with tools such as Ansible, Terraform and Kubernetes.
- **Use cases**: Hybrid cloud, container orchestration and automation.

Advantages

- **Flexibility**: Supports multiple hypervisors, storage and networking options.
- **Ease of use**: Simple and intuitive interface for managing cloud resources.
- **Integration**: Works with existing IT infrastructure and tools.
- **Scalability**: Designed to scale from small deployments to large data centres.

2.4. EUCALYPTUS

Eucalyptus (Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems) is a powerful and flexible open-source cloud computing software platform that

provides infrastructure as a service (IaaS) and offers a wide range of features for managing virtualized data centre resources. It is designed to be compatible with Amazon Web Services (AWS) APIs, which makes it an attractive option for organizations that want to build private clouds that can be integrated seamlessly with AWS. Eucalyptus is widely used by enterprises, service providers and research institutions to build scalable and flexible cloud environments. Eucalyptus consists of several interconnected components, each of which is responsible for a specific aspect of the cloud infrastructure.

Core functions and components

1. Cloud Controller (Cloud Controller-CLC)

- **Purpose**: Front-end component that provides the web-based user interface and API endpoints.
- **Key features**:
  - **User interface**: Web-based dashboard for managing cloud resources.
  - **API endpoints**: Supports AWS-compatible APIs (EC2, S3, IAM, etc.).
  - **Resource management**: Manages cloud resources and services at a high level.
- **Use cases**: Cloud management, user authentication and API access.

2. Cluster Controller (Cluster Controller-CC)

- **Purpose**: Manages the life cycle of virtual machines within a cluster.
- **Key features**:
  - **Virtual machine management**: Creating, deploying, cloning, migrating and terminating virtual machines.
  - **Resource scheduling**: Allocates compute resources based on availability and defined policies.
  - **Network management**: Manages network connectivity and IP addresses for virtual machines.
- **Use cases**: Provisioning, scaling and managing virtual machines.

3. Node Controller (Node Controller-NC)

- **Purpose**: Manages the physical compute nodes that host virtual machines.
- **Key features**:
  - **Hypervisor support**: Works with KVM, Xen and VMware.
  - **Virtual machine life cycle management**: Starting, stopping and monitoring virtual machines.
  - **Resource monitoring**: Tracking of processor, memory, disk and network utilization.
- **Use cases**: Hosting virtual machines, resource monitoring and hypervisor management.

4. Storage Controller (Storage Controller-SC)

- **Purpose**: Manages data storage resources for virtual machines and the cloud infrastructure.

- **Key features**:
  - **Elastic Block Store (EBS)**: Provides persistent block storage volumes for virtual machines.
  - **Snapshots**: Creates and manages volume snapshots for data backup and recovery.
  - **Storage Gateway**: A storage gateway that integrates with on-premises storage solutions.
- **Use cases**: Virtual machine storage, data backup and block storage management.

5. Walrus

- **Purpose**: Provides S3-compatible object storage for unstructured data.
- **Key features**:
  - **Scalable storage**: Stores and retrieves large amounts of unstructured data.
  - **Resilience**: Replicates data across multiple nodes for high availability.
  - **RESTful API**: Enables programmatic access to stored objects.
- **Use cases**: Backup storage, media repositories and data archiving.

6. Identity and access management (Identity and Access management-IAM)

- **Purpose**: Provides authentication, authorization and resource isolation.
- **Key features**:
  - **User management**: Creates and manages users and groups.
  - **Role-based access control (RBAC)**: Defines roles and permissions for users.
  - **Compatibility with AWS IAM**: Supports AWS IAM policies and roles.
  - **LDAP/AD integration**: Works with LDAP and Active Directory for user authentication purposes.
- **Use cases**: User management, secure access control and multi-tenant environments.

7. Networking

- **Purpose**: Manages network connectivity and IP addresses for cloud resources.
- **Key features**:
  - **Elastic IP addresses**: Assigns and manages static IP addresses of virtual machines.
  - **Security groups**: Defines firewall rules for virtual machines.
  - **Virtual private cloud (VPC)**: Creating isolated virtual networks.
  - **Load balancing**: Distribution of incoming traffic across multiple virtual machines.
- **Use cases**: Network isolation, secure communication and traffic management.

8. Management and monitoring

- **Purpose**: Provides tools for managing and monitoring cloud resources.
- **Key features**:

  - **Web-based information panel (dashboard)**: User-friendly interface for managing cloud resources.
  - **Resource monitoring**: Tracking of processor, memory, disk and network utilization.
  - **Usage reporting**: Creating reports for billing and chargeback.
  - **Alerts**: Sets alerts regarding resource utilization and system performance metrics.
- **Use cases**: Cloud management, monitoring and billing.

9. High availability and scalability

- **Purpose**: Ensures service continuity and scalability.
- **Key features**:
  - **High availability (High Availability)**: Redundant components and failover mechanisms.
  - **Scalability**: Designed to scale from small deployments to large data centres.
  - **Load balancing**: Distributes the load across multiple hosts.
- **Use cases**: High availability, disaster recovery and scalable cloud infrastructure.

10. Security

- **Purpose**: Ensures secure access to and operation of cloud resources.
- **Key features**:
  - **Network isolation**: VLANs and private networks for secure communication.
  - **Encryption**: Supports SSL/TLS for secure access to the API and the user interface.
  - **Access control**: Fine-grained permissions for users and resources.
  - **Security groups**: Defines firewall rules for virtual machines.
- **Use cases**: Secure cloud environments and compliance.

11. Ecosystem integration

- **Purpose**: Integrates with existing IT infrastructure and tools.
- **Key features**:
  - **Cloud Bursting**: Integration with public clouds such as AWS for hybrid cloud deployment.
  - **Third-party tools**: Works with tools such as Ansible, Terraform and Kubernetes.
  - **API and CLI**: Comprehensive API and command-line tools for automation and integration.
- **Use cases**: Hybrid cloud, automation and integration with existing IT infrastructure.

Advantages

- **AWS compatibility**: Seamless integration with AWS APIs and services.
- **Flexibility**: Supports multiple hypervisors, storage and networking options.

- **Ease of use**: Simple and intuitive interface for managing cloud resources.
- **Scalability**: Designed to scale from small deployments to large data centres.

2.5. COMPARISON OF IAAS PLATFORMS

Functionality	Apache CloudStack	OpenStack	OpenNebula	Eucalyptus
Overview				
Service type	IaaS	IaaS	IaaS	IaaS
Primary use case	Public, private and hybrid clouds	Public, private and hybrid clouds	Private and hybrid clouds	Private and hybrid clouds (AWS-compatible)
Licence	Apache 2.0 licence	Apache 2.0 licence	Apache 2.0 licence	GPLv3
Community	Strong community with enterprise support	Largest and most active cloud community	Smaller but active community	Smaller community focused on AWS compatibility
Architecture				
Modularity	Monolithic architecture with tightly integrated components	Highly modular with loosely coupled components	Modular, but more integrated components than OpenStack	Modular with AWS-compatible components
Components	Single management server with agents on hosts	Multiple independent services (Nova, Neutron, Cinder, etc.)	Centralized management with plug-ins for storage, networking, etc.	AWS-compatible services (EC2, S3, IAM, etc.)
Scalability	Good for large deployments	Very good, but requires careful configuration	Good for small and medium deployments	Good for small and medium deployments
Hypervisor support				
Supported hypervisors	KVM, VMware, XenServer, Hyper-V, LXC	KVM, VMware, Xen, Hyper-V, LXC	KVM, VMware, Xen, LXD	KVM, Xen, VMware
Bare-Metal support	Limited	Yes (Ironic)	Yes (using Bare-Metal drivers)	Limited
Storage management				
Block storage	Yes (using plug-ins for NFS, iSCSI, Ceph, etc.)	Yes (Cinder)	Yes (support for multiple backends)	Yes (Elastic Block Store)
Object storage	Limited (using plug-ins)	Yes (Swift)	Limited	Yes (Walrus, S3-compatible)
File storage	Limited	Yes (Manila)	Yes (using plug-ins)	Limited
Snapshots	Yes	Yes	Yes	Yes
Networking				
Network model	Basic and advanced	Highly flexible (Neutron)	Flexible with support for	AWS-compatible networking

	networking modes		VLANs and private networks	(VPC, Security Groups)
SDN support	Yes (using plug-ins)	Yes (Open vSwitch, Cisco ACI, etc.)	Yes (Open vSwitch)	Limited
Load balancing	Yes	Yes (using Neutron LBaaS)	Yes	Yes
Firewall	Yes	Yes (security groups)	Yes (security groups)	Yes (security groups)
User management				
Multi-tenancy	Yes	Yes	Yes	Yes
RBAC	Yes	Yes (using Keystone)	Yes	Yes (using IAM)
LDAP/AD integration	Yes	Yes	Yes	Yes
API and extensibility				
API	RESTful API	RESTful API	RESTful API	AWS-compatible API (EC2, S3, IAM)
Extensibility	Yes (using plug-ins)	High extensibility (using plug-ins and drivers)	Yes (using plug-ins)	Limited (focused on AWS compatibility)
CLI tools	Yes	Yes	Yes	Yes
High availability/scalability				
High availability	Yes (virtual machine failover, redundant management servers)	Yes (using redundant components)	Yes (virtual machine failover, high availability settings)	Yes (redundant components)
Scalability	Good for large deployments	Very good, but requires careful configuration	Good for small and medium deployments	Good for small and medium deployments
Security				
Network isolation	Yes (VLANs, private networks)	Yes (VLAN, VXLAN, etc.)	Yes (VLANs, private networks)	Yes (VPC, security groups)
Encryption	Yes (SSL/TLS)	Yes (SSL/TLS)	Yes (SSL/TLS)	Yes (SSL/TLS)
Access control	Yes (RBAC)	Yes (RBAC using Keystone)	Yes (RBAC)	Yes (RBAC using IAM)
Use cases				
Public cloud	Yes	Yes	Limited	Limited
Private cloud	Yes	Yes	Yes	Yes
Hybrid cloud	Yes	Yes	Yes	Yes (AWS-compatible)
Development/test environments	Yes	Yes	Yes	Yes
Disaster recovery	Yes	Yes	Yes	Yes
Strengths and weaknesses				

Strengths	- simple deployment and management - strong enterprise support - scalable	- highly modular and flexible - largest community - extensive functionality	- ease of use - lightweight - suitable for private clouds	- AWS compatibility - simple deployment - suitable for hybrid clouds
Weaknesses	- less modular than OpenStack - smaller community than OpenStack's	- complex deployment and management - steeper learning curve	- limited public cloud features - smaller community	- limited to AWS compatibility - smaller community

2.6. EVALUATION

Apache CloudStack: The best solution for enterprises seeking a simple, scalable and easily manageable private or hybrid cloud solution.

OpenStack: An ideal solution for organizations that need a highly flexible and modular cloud platform with extensive functionality.

OpenNebula: An excellent solution for small and medium-sized private clouds, especially for organizations that prioritize ease of use and easy deployment.

Eucalyptus: An ideal solution for organizations seeking AWS-compatible private or hybrid cloud solutions.

3. CONTAINER ORCHESTRATION SYSTEMS

3.1. KUBERNETES (K8S)

Kubernetes is the de facto standard for container orchestration, enabling organizations to efficiently build, deploy and manage modern cloud applications. K8s is a very powerful tool for managing containerized applications.

Core functions and components

1. Container orchestration

Kubernetes automates the deployment, scaling and management of containerized applications. It groups containers into logical units called **Pods** (the smallest deployable units in Kubernetes) and manages their life cycle within a cluster of machines.

- **Pods**: A Pod is a group of one or more containers that share storage, network and specifications. Containers in a Pod are always co-located and co-scheduled.
- **Cluster**: A Kubernetes cluster consists of the **control plane** (master nodes) and **worker nodes**. The control plane manages the cluster, while the worker nodes run the applications.

2. Service discovery and load balancing

Kubernetes automatically assigns IP addresses and DNS names to Pods, making it easier for applications to find and communicate with each other.

- **Services**: A Kubernetes Service is an abstraction that defines a logical set of Pods and a policy for accessing them. It provides stable IP addresses and DNS names.
- **Load Balancing**: Kubernetes can distribute network traffic to Pods using built-in load-balancing tools or integrate with cloud providers' load-balancing tools.

3. Storage orchestration

Kubernetes enables storage systems (local, cloud or network) to be dynamically attached to containers.

- **Persistent volumes (Persistent Volume PV)**: Provides persistent storage for stateful applications (e.g. databases).
- **Persistent volume claims (Persistent Volume Claim PVC)**: Enables users to request storage without concern for the underlying infrastructure.

4. Self-healing

Kubernetes ensures high availability through automatic recovery from failures.

- **Health checks**: Kubernetes uses **liveness probes** to restart containers that fail and **readiness probes** to ensure that containers are ready to serve traffic.
- **Automatic replacement**: If a node fails, Kubernetes reschedules Pods onto healthy nodes.

5. Automated rollout (rollout) and rollback (rollback)

Kubernetes supports declarative application updates, ensuring smooth rollouts and rollbacks.

- **Deployments**: A Deployment object enables the desired state of an application to be defined. Kubernetes automatically handles rolling updates and rollbacks.

- **Canary deployment**: Enables changes to be rolled out gradually to a subset of users in order to test new features.
- **Rollbacks**: If an update fails, Kubernetes can revert to the previous stable version.

6. Horizontal scaling

Kubernetes can automatically scale applications based on resource utilization or custom metrics.

- **Horizontal Pod autoscaler (Horizontal Pod Autoscaler HPA)**: Scales the number of Pods based on CPU, memory or custom metrics.
- **Manual scaling**: Applications can be scaled manually.

7. Configuration and secrets management

Kubernetes provides tools for managing configuration data and sensitive information.

- **ConfigMaps**: Enables configuration data to be stored as key-value pairs, separately from container images.
- **Secrets**: Ensures secure storage of sensitive information such as passwords, tokens or SSH keys.

8. Batch processing and CI/CD workloads

Kubernetes supports both long-running applications and batch jobs.

- **Jobs**: Running tasks to completion (e.g. data processing).
- **CronJobs**: Scheduling of periodic tasks (e.g. backups).
- **CI/CD integration**: Kubernetes integrates with CI/CD tools such as Jenkins, GitLab and ArgoCD for automated deployments.

9. Multi-tenancy and namespaces

Kubernetes provides logical isolation for teams, projects or environments.

- **Namespaces**: Enables cluster resources to be divided into virtual clusters (e.g. "dev", "prod").
- **Resource quotas**: Used to limit resource usage per namespace in order to prevent excessive consumption.

10. Extensibility

Kubernetes is highly extensible and enables users to customize its behaviour.

- **Custom resource definitions (Custom Resource Definitions CRDs)**: Enables custom resources to be defined to extend the Kubernetes API.
- **Operators**: Enables the use of K8s operators to automate application management (e.g. databases, monitoring tools).

11. Portability and cloud agnostic

Kubernetes works in on-premises, hybrid and multi-cloud environments.

- **Cloud provider integration**: Works seamlessly with AWS, Azure, GCP and others.
- **Hybrid cloud**: Enables workloads to be run in on-premises and cloud environments.

12. Monitoring and logging

Kubernetes provides tools and integrations for monitoring and logging.

- **Metrics server**: Collects resource utilization data (CPU, memory).
- **Prometheus**: A popular monitoring tool for Kubernetes.
- **Logging**: Integrates with logging solutions such as Fluentd, Elasticsearch and Kibana.

13. Network management

Kubernetes manages networking between Pods, Services and external systems.

- **Network policies**: Define rules for communication between Pods.
- **Ingress**: Manages external access to Services, typically via HTTP/HTTPS.

14. Resource management

Kubernetes allocates and manages resources efficiently.

- **Requests and limits**: Used to define CPU and memory requests and limits for individual containers.
- **Quality of Service (QoS)**: Ensures that critical workloads receive the necessary resources.

15. Security

Kubernetes provides robust security features to protect clusters and workloads.

- **Role-based access control (RBAC)**: Enables fine-grained permissions to be defined for users and services.
- **Network rules**: Used to restrict traffic between individual Pods.
- **Secrets management**: Enables sensitive data to be stored and managed securely.
- **Pod security policies (Pod Security Policies)**: Used to enforce Pod security policies.

16. Declarative configuration

Kubernetes uses YAML or JSON files to define the desired state of the system.

- **Declarative model**: Once the desired values have been defined, Kubernetes ensures that the system matches this state.
- **GitOps**: Enables Kubernetes configuration to be managed using Git repositories.

17. Cluster management

Kubernetes simplifies cluster setup, upgrades and maintenance.

- **kubeadm**: A tool for bootstrapping Kubernetes clusters.
- **Cluster API**: A Kubernetes project for declarative cluster management.
- **Upgrades**: Kubernetes supports rolling upgrades with minimal downtime.

Advantages

- **Scalability**: Handles applications of any size, from small teams to enterprise systems.
- **Resilience**: Self-healing and high availability ensure that applications remain in operation.
- **Portability**: Enables applications to be run anywhere, from on-premises to multi-cloud environments.
- **Ecosystem**: An extensive ecosystem of tools, plug-ins and integrations (e.g. Helm, Istio, Prometheus).

3.2. DOCKER SWARM

Docker Swarm is a powerful yet simple tool for orchestrating and managing containerized applications across multiple Docker hosts.

Core functions and components

1. Cluster management

Docker Swarm turns a group of Docker hosts into a single virtual system, enabling them to be managed as a unified cluster.

Key components:

Nodes:

- **Manager nodes**: Used for cluster management, orchestration and scheduling. These nodes maintain the cluster state and manage the worker nodes.
- **Worker nodes**: Responsible for actually running the current container workload. These nodes receive tasks from the manager nodes.
- **High availability**: Manager nodes use the **Raft consensus algorithm** to ensure fault tolerance. If the leading manager fails, another manager takes over its role.

2. Service orchestration

Docker Swarm enables the definition and management of **services**, which are the basic abstraction for running applications.

Key features:

- **Services**: A service is a definition of tasks to be executed on worker nodes (e.g. running a web server or a database).
- **Replicas**: Enables the number of replicas (identical copies) of a specific service to be run to be specified.
- **Global mode**: Enables one instance of a service to be run on every node in the cluster (e.g. for monitoring agents).

3. Declarative service model

Docker Swarm uses a declarative approach to define the desired state of services.

Key features:

- **Stack files**: Using "Docker Compose" YAML files, multi-service applications can be defined, e.g.:

```
version: "3.8"
  services:
    web:
      image: nginx
      ports:
        - "80:80"
      deploy:
        replicas: 3
```

- **Desired state**: Docker Swarm continuously ensures that the cluster matches the desired state defined in the stack file.

4. Rolling updates and rollbacks

Docker Swarm supports zero-downtime updates and service rollbacks.

Key features:

- **Rolling updates**: (Rolling updates) Enables individual containers to be gradually replaced with new versions in order to ensure uninterrupted operation.
- **Rollbacks**: If an update fails, enables a return to the previous version of the service.

5. Network management

Docker Swarm provides robust networking features that enable communication between containers and services.

Key features:

- **Overlay networks**: Enables secure multi-host networks to be created for containers for the purpose of communication between nodes.
- **Service discovery**: Enables DNS names to be assigned to services automatically and workloads to be balanced.
- **Ingress network**: (Ingress Network) Used to route external traffic to the appropriate service.

6. Security

Docker Swarm includes several security features to protect the cluster and workloads.

Key features:

- **Mutual TLS**: Used to encrypt communication between individual nodes using mutual TLS.
- **Secrets management**: Enables sensitive data (e.g. passwords, API keys) to be stored and managed securely using Docker Secrets.
- **Role-based access control (RBAC)**: Enables restricted access to be set for the manager nodes of the Docker Swarm administrator.

7. Resource management

Docker Swarm enables allocated resources to be managed efficiently.

Key features:

- **Resource constraints**: Enables processor and memory limits to be set for individual services.
- **Reservation**: Used to reserve the necessary resources for specific services.

8. Multi-host scheduling

Docker Swarm automatically schedules individual containers across nodes based on resource availability and individual constraints.

Key features:

- **Scheduling**: Used for the even distribution of tasks among individual worker nodes.
- **Constraints**: Enables placement constraints to be defined (e.g. allows containers to be run only on nodes with specific labels).

9. Ease of use

Docker Swarm is designed to be simple and easy to use, especially for teams that already have experience with Docker.

Key features:

- **Built-in support**: No additional installation is required – it is enough simply to enable Swarm mode.
- **Docker CLI**: Used to manage Swarm clusters using Docker commands on the command line.

10. Monitoring and logging

Docker Swarm provides tools and integrations for monitoring and logging system messages.

Key features:

- **Docker logs**: Enables access to service and container logs.
- **Third-party integrations**: Enables integration with monitoring tools such as Prometheus and logging solutions such as the ELK Stack.

11. Fault tolerance

Docker Swarm ensures high availability and resilience of individual applications.

Key features:

- **Self-healing**: Docker can automatically reschedule individual containers in the event of a node failure.
- **Replication**: Ensures that the required number of replicas of a specific service is always running.

12. Portability

Docker Swarm is highly portable and works in various environments.

Key features:

- **Docker Compose compatibility**: Enables existing Docker Compose files to be used to deploy applications in Swarm mode.
- **Cross-Platform**: Works on any platform that supports Docker.

13. Extensibility

The functionality of Docker Swarm can be extended using plug-ins and custom configurations (e.g. volume plug-ins, network plug-ins).

Advantages

- **Ease of use**: Docker Swarm offers simple usage.
- **Scalability**: Docker Swarm is usable for small to medium-sized workloads.
- **Setup**: Docker Swarm enables quick and simple system setup.

3.3. APACHE MESOS

Apache Mesos is a powerful tool for managing distributed systems, designed to abstract and manage resources in a cluster of machines. It is particularly suitable for running

various workloads, such as big data processing, microservices and various batch jobs, on the same cluster.

Core functions and components

1. Architecture

Apache Mesos follows a master-slave architecture with a two-level scheduling model.

Key components:

- **Mesos Master**:
  - Its task is to manage cluster resources and offer them to individual frameworks.
  - It uses ZooKeeper for leader election and to ensure high availability.
- **Mesos agents (slaves)**:
  - Agents run on worker nodes and execute the individual tasks of frameworks.
  - They are used to report information on the availability of individual resources (CPU, memory, etc.) to the master server. -**Framework**:
  - They represent applications or services that run on Mesos (e.g. Hadoop, Spark, Marathon).
  - They consist of a **scheduler** (for accepting/declining resource offers) and an **executor** (for running tasks).
- **Zookeper**:
  - Its role is to provide distributed coordination and leader election for the Mesos master.

2. Resource management

Mesos abstracts and manages resources within the cluster, enabling efficient sharing and isolation.

Key features:

- **Resource abstraction**:
  - Mesos can abstract individual resources such as CPU, memory, storage and others into a unified pool.
- **Fine-grained resource allocation ** (Fine-grained Allocation):
  - Mesos enables resources to be allocated at a fine-grained level (e.g. 0.1 CPU core, 128 MB of memory).
- **Dynamic resource allocation**:
  - Enables resources to be dynamically reallocated based on workload demands.

3. Scalability

Mesos is designed to handle large clusters and workloads.

Key features:

- **Large clusters**:
  - It can manage tens of thousands of nodes and millions of tasks.

- **Efficient resource utilization**:
    - Maximizes resource utilization by dynamically allocating resources to individual frameworks.

4. Multi-framework support

Mesos supports multiple frameworks simultaneously, which makes it ideal for heterogeneous workloads.

Key features:

- **Framework Agnostic**:
    - Enables multiple frameworks (e.g. Hadoop, Spark, Kafka, Marathon) to be run on the same cluster.
- **Custom frameworks**:
    - Developers can create custom frameworks for specific workloads using the Mesos API.

5. Fault tolerance

Mesos ensures high service availability and resilience for distributed applications.

Key features:

- **High availability (HA)**:
    - Uses ZooKeeper for leader election and fault tolerance. -**Self-healing (Self-healing)**:
    - The system is able to recover automatically from a failure, thereby ensuring high availability.

6. Isolation

Mesos provides resource isolation so as to ensure that the workloads of individual resources do not interfere with each other.

Key features:

- **Resource isolation**:
    - The system uses Linux-type containers (e.g. Docker, cgroups) to isolate individual resources between tasks.
- **Multi-tenancy**:
    - The system supports multi-tenant environments with secured resource isolation.

7. Scheduling

Mesos uses a **two-level scheduling model** for efficient resource allocation.

Key features:

- **Two-level scheduling**:
    - **Mesos Master**: Its task is to provide system resources to individual frameworks.
    - **Frameworks**: Accept or decline the offered resources based on their needs.

- **Dynamic workloads**:
    - The system is able to handle dynamic workloads efficiently by reallocating resources according to needs.

8. Extensibility

Mesos is highly extensible, which enables users to customize its functionality.

Key features:

- **Modular architecture**:
    - The system can easily be extended using custom allocators, isolators and frameworks.
- **API support**:
    - The system provides APIs for developing custom frameworks and for integration with existing systems.

9. Monitoring and metrics

Mesos provides tools and integrations for monitoring tools and metrics.

Key features:

- **Built-in metrics**:
    - The system enables individual cluster and framework metrics to be tracked, such as CPU, memory and disk utilization, and others.
      -**Integration**:
    - The system enables full integration with various monitoring tools such as Prometheus or Grafana.

10. Portability

Mesos is very adaptable and works in various environments.

Key features:

- **Multi-platform**:
    - Mesos can run on Linux, macOS and Windows.
- **Cloud and On-Premise**:
    - The system can operate in cloud, on-premises and hybrid environments.

11. Ecosystem

Mesos has a rich ecosystem of frameworks and tools for various workloads.

Key frameworks:

-**Marathon**:

- Used for long-running services and container orchestration.
  -**Chronos**:
- Used for job scheduling (similar to cron).
- **Apache Spark**:
    - Used for distributed data processing.
- **Apache Kafka**:

- Used for distributed streaming.

12. Security

Mesos provides robust security features to protect the cluster and workloads.

Key features:

-**Authentication**:

- The system supports various types of authentication for individual frameworks and agents.

-**Authorization**:

- The system provides fine-grained access control to resources.

-**Encryption**:

- Communication between the individual components is encrypted.

Advantages

- **Resource management** – A multi-level system that allows fine granularity.
- **Scalability** - The system is designed for large clusters.
- **Ecosystem** - A rich ecosystem for big-data needs

3.4. HASHICORP NOMAD

HashiCorp Nomad is a flexible and powerful workload orchestration tool designed for deploying and managing applications and containers in on-premises and cloud environments. It is known for its simplicity, scalability and support for various workloads, including containers, virtual machines and standalone applications.

Core functions and components

1. Workload orchestration

Nomad abstracts the infrastructure and provides a unified platform for deploying and managing workloads.

Key features:

- **Multi-cloud and hybrid cloud**: Nomad can operate in on-premises, cloud and hybrid environments.
- **Diverse workloads**: Various kinds of containers (Docker, Podman), virtual machines (QEMU, Firecracker) and standalone applications (Java, Python, binaries) are supported.
- **Declarative job specification**: It allows a workload to be defined using a simple, human-readable job file (in HCL or JSON format).

2. Scalability

Nomad is designed to handle clusters of any size, from small teams to large enterprises.

Key features:

- **Horizontal scaling**: The system allows applications to be easily scaled up or down by modifying the job file.

- **High performance**: The system can handle millions of containers and thousands of nodes with low latency.

3. Ease of use

Nomad is known for its simplicity and minimal operational costs.

Key features:

- **Single Binary**: Nomad is distributed as a single binary file, which allows easy installation.
- **No dependencies**: It does not require external databases or consensus systems (optional integration with Consul for service discovery is available).
- **Intuitive CLI and user interface**: Nomad provides a command-line interface (CLI) and a web user interface (UI) for managing workloads.

4. Scheduling

Nomad uses a sophisticated scheduler to allocate workloads to the most suitable nodes.

Key features:

- **Bin Packing**: It efficiently packs jobs onto nodes in order to maximize resource utilization.
- **Affinity and Anti-Affinity**: The system allows control over job placement based on the attributes of a node or of other jobs.
- **Dynamic workloads**: The system handles workloads dynamically and redistributes individual resources as needed.

5. Multi-datacentre and multi-region support

Nomad is designed for global deployment across multiple data centres and regions.

Key features:

- **Federated clusters**: The system allows a single Nomad cluster to run across multiple data centres or regions.
- **Cross-datacentre scheduling**: The system allows workloads to be scheduled in specific data centres or regions.

6. Fault tolerance and self-healing

Nomad provides high availability and resilience for individual workloads.

Key features:

- **High availability**: The system supports multiple server nodes for increased fault tolerance.
- **Self-healing**: The system can automatically restart failed jobs and reschedule them onto healthy nodes.
- **Health-checked restarts**: The system allows automatic restarting of unhealthy jobs based on health checks of the individual jobs.

7. Resource management

Nomad allows resources to be efficiently managed and allocated within a given cluster.

Key features:

- **Resource allocation**: The system allocates CPU, memory, disk and network resources to individual jobs.
- **Resource isolation**: The system uses cgroups, namespaces and other isolation mechanisms to ensure that jobs do not interfere with one another.

8. Integrations

Nomad integrates seamlessly with other HashiCorp tools and third-party systems.

Key features:

- **Integration with Consul**: Provides service discovery, health checking and networking.
- **Integration with Vault**: Allows secrets and sensitive data to be managed securely.
- **Integration with Terraform**: Allows infrastructure provisioning and Nomad cluster setup to be automated.

9. Monitoring and observability

Nomad provides tools and integrations for monitoring and observability.

Key features:

- **Built-in metrics**: The system allows cluster and job metrics (e.g. CPU, memory usage) to be tracked.
- **Logging**: The system allows job logs to be collected and streamed.
- **Integration**: The system can work with monitoring tools such as Prometheus, Grafana and Datadog.

10. Security

Nomad includes robust security features to protect the cluster and workloads.

Key features:

- **ACL (Access Control Lists)**: The system allows access control to individual resources based on roles and policies.
- **mTLS (Mutual TLS)**: Communication between Nomad components is encrypted.
- **Integration with Vault**: Used for the secure management of secrets and sensitive data.

11. Extensibility

Nomad is highly extensible and allows users to customize its behaviour according to their own needs.

Key features:

- **Plugins**: The system can be extended with plugins for task drivers, devices and more.
- **API**: Provides APIs for integration with custom tools and workflows.

Advantages

- **Simplicity** - The system is a simple and lightweight orchestrator with minimal operational overhead.
- **Diverse workloads** - The system allows various kinds of workloads (containers, virtual machines, standalone applications) to be run on the same platform.

- **Multi-datacentre or multi-region support**.

3.5. COMPARISON OF CONTAINER ORCHESTRATION TOOLS

Functionality	Kubernetes	Docker Swarm	Apache Mesos	HashiCorp Nomad
Overview				
Primary use case	Container orchestration for complex and large-scale workloads	Simple container orchestration for small to medium workloads	Resource management and scheduling of diverse workloads (e.g. big data, containers)	Workload orchestration for various types (containers, virtual machines, standalone applications)
Ease of use	Steeper learning curve	Simple and lightweight	Medium complexity	Simple and lightweight
Ecosystem	Extensive ecosystem and integrations	Limited ecosystem	Rich ecosystem for big data	Integration with HashiCorp tools
Scalability	High scalability	Best for small to medium clusters	High scalability	High scalability
Architecture				
Architecture	Master-slave nodes (control plane and worker nodes)	Manager-worker nodes (Swarm mode)	Master-slave nodes (with ZooKeeper for HA)	Server-client nodes (optional high availability with multiple servers)
Scheduling model	Centralized scheduler	Decentralized scheduling	Two-level scheduling (resource offers)	Centralized scheduler
High availability	Built-in (etcd for state storage)	Built-in (Raft consensus)	Requires ZooKeeper for HA	Built-in (Raft consensus)
Workload support				
Containers	Yes (Docker, containerd, CRI-O)	Yes (Docker only)	Yes (Docker, AppC)	Yes (Docker, Podman)
Virtual machines	Limited (using KubeVirt)	No	Yes (using frameworks such as Marathon)	Yes (QEMU, Firecracker)
Standalone applications	No	No	Yes (via custom executors)	Yes (Java, Python, executables)
Batch jobs	Yes (Jobs, CronJobs)	Limited	Yes (using Chronos)	Yes (batch and periodic jobs)
Resource management				
Resource allocation	CPU, memory, storage	CPU, memory	CPU, memory, storage, ports	CPU, memory, disk, network
Resource isolation	Namespaces, cgroups	Namespaces, cgroups	Containers, groups	Namespaces, cgroups
Multi-tenancy	Yes (namespaces, RBAC)	Limited	Yes (roles, quotas)	Yes (ACLs, namespaces)

Network management				
Service discovery	Built-in (DNS, kube-dns, CoreDNS)	Built-in (DNS-based)	Framework-dependent (e.g. Marathon)	Integration with Consul
Load balancing	Built-in (kube-proxy, Ingress)	Built-in	Framework-dependent	Integration with Consul
Network policies	Yes (fine-grained traffic control)	Limited	Framework-dependent	Limited
Data storage				
Persistent storage	Yes (persistent volumes, PVC)	Limited	Yes (persistent volumes)	Yes (CSI plugins)
Dynamic provisioning	Yes	No	Yes	Yes
Fault tolerance and self-healing				
Self-healing	Yes (restarts failed containers)	Yes (restarts failed tasks)	Yes (depending on the framework)	Yes (restarts failed tasks)
High availability	Yes (etcd, multi-master)	Yes (Raft consensus)	Yes (ZooKeeper)	Yes (Raft consensus)
Monitoring and observability				
Built-in metrics	Yes (metrics server)	Limited	Yes	Yes
Logging	Yes (Fluentd, Elasticsearch)	Limited	Framework-dependent	Yes (integrated logging)
Integration	Prometheus, Grafana, etc.	Limited	Prometheus, Grafana, etc.	Prometheus, Grafana, etc.
Security				
Authentication	Yes (RBAC, OIDC)	Yes (TLS, mutual TLS)	Yes (authentication modules)	Yes (ACLs, Vault integration)
Authorization	Yes (RBAC)	Limited	Yes (tasks, quotas)	Yes (ACL)
Secrets management	Yes (Secrets)	Yes (Docker Secrets)	Framework-dependent	Yes (integration with Vault)
Extensibility				
Custom resources	Yes (CRD)	No	Yes (custom frameworks)	Yes (plugins, APIs)
Plugins	Yes (CNI, CSI, etc.)	Limited	Yes (isolators, allocators)	Yes (task drivers, device drivers)
Use cases				
Microservices	Excellent	Good	Good (with Marathon)	Excellent
Big data	Limited	No	Excellent (Hadoop, Spark, Kafka)	Good (with integrations)
Batch processing	Yes (Jobs, CronJobs)	Limited	Yes (Chronos)	Yes (batch and periodic jobs)
Legacy applications	Limited	No	Yes (custom executors)	Yes (standalone applications)

Strengths and weaknesses				
Strengths	- extensive ecosystem - highly scalable - rich feature set	- simple and lightweight - easy setup - integration with Docker	- support for multiple frameworks - scalability - big-data integration	- simple and lightweight - support for multiple workload types - scalable
Weaknesses	- steeper learning curve - complex setup and management	- limited ecosystem - less suitable for large clusters	- medium complexity - requires additional tools for full functionality	- smaller ecosystem - less mature compared with Kubernetes

3.6. EVALUATION

Kubernetes: The best solution for large-scale and complex workloads requiring a rich ecosystem and advanced features.

Docker Swarm: An ideal solution for small and medium-sized teams seeking simplicity and integration with Docker.

Apache Mesos: A great solution for heterogeneous workloads, especially big data and multi-architecture environments.

HashiCorp Nomad: An ideal solution for teams seeking simplicity, support for multiple workloads and integration with HashiCorp tools.