PROJECT

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

V12 Technological architecture

KPB4 – Solution architecture

type – documentation

month of submission – M22

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

Contents

1. Introduction 3
2. Application architecture 3
1.1. Application Viewpoints 4
3. Technological architecture 8
1.2. Design of the technological architecture 8
1.1.1. Technology Viewpoint CTI 9
1.1.2. Technology Viewpoint EDR/XDR 18
1.1.3. Technology Viewpoint INNOSOC Common 23
1.1.4. Technology Viewpoint Asset inventory 26
1.1.5. Technology Viewpoint Kubernetes 30
1.1.6. Technology Viewpoint OpenStack 54
1.1.7. Technology Viewpoint OpenStack External 60
1.1.8. Technology Viewpoint Reporting 61
1.1.9. Technology Viewpoint SIEM 64
1.1.10. Technology Viewpoint Vulnerability scanner 68
1.1.11. Technology Viewpoint SOAR 72
1.1.12. Technology Viewpoint Ticketing system/ Incident recording and management 78
1.1.13. Technology Viewpoint Knowledge portal for the SOC team 82

INTRODUCTION

The aim of the project is to develop a technologically modern, open and scalable solution that will be capable of supporting organizations in effectively dealing with cyber threats.

1. APPLICATION ARCHITECTURE

The application architecture of the solution represents the logical design of the application services, components and their mutual links that provide the functionality of the proposed SOCaaS platform. Its purpose is to provide a comprehensive view of the application layer of the solution, in particular in terms of the division of responsibilities between the individual components, their functional focus and the way in which they jointly support the main processes of security monitoring, detection, evaluation and handling of security events and incidents. The architecture is based on the model developed in the EA tool, with the descriptions of the individual objects derived from the notes assigned to the modelled elements.

The proposed application architecture reflects the nature of the solution as a multi-tenant cybersecurity operations centre service. From the perspective of the application layer, it covers in particular the areas of security event collection and correlation, response automation, incident management, threat intelligence, asset management, vulnerability management, reporting and support for knowledge management of the SOC team. The application layer defined in this way at the same time creates the precondition for supporting key use cases, such as security event monitoring, detection rule management, incident escalation and handling, threat intelligence management, asset management, vulnerability management, report export and management of users, roles and organizations within the SOCaaS platform.

In the document, the application architecture is presented as a high-level list of the application components of the solution. The application viewpoints diagram shows the main components that together form the functional basis of the platform. The available model shows that the core of the solution consists mainly of the components SIEM, SOAR, CTI, EDR/XDR, the analytics and automation engine, incident recording and management, the ticketing system, cybersecurity incident management, asset inventory, the vulnerability scanner, reporting and the knowledge portal for the SOC team. The diagram therefore does not represent a detailed design of internal integration flows, but above all an overview of the main application building blocks of the solution.

From a functional point of view, the individual application components can be divided into several logical groups. The first group consists of components oriented towards collection, analysis and detection, i.e. mainly SIEM, EDR/XDR, CTI and the analytics and automation engine. These components ensure the receipt and processing of security data, their enrichment with threat context, event correlation and the identification of anomalies or suspicious states. The second group consists of components oriented towards response and incident life cycle management, primarily SOAR, incident recording and management, the ticketing system and cybersecurity incident management. Their role is to enable systematic recording, coordination, automation and management of incident response from the moment incidents arise until they are closed.

A further area of the application architecture consists of supporting and contextual components, in particular asset inventory, the vulnerability scanner, reporting and the knowledge portal for the SOC team. Asset inventory provides an overview of the managed devices, systems and resources, thereby creating the basic context for monitoring and assessing the impacts of incidents. The vulnerability scanner supports a proactive approach to security and provides inputs for the prioritization of corrective measures. Reporting ensures the aggregation, visualization and distribution of security information to various target groups. The knowledge portal serves as a central place for storing playbooks, analytical procedures, recommendations and experience from incident handling.

A special place in the application architecture is held by the analytics and automation engine, which, according to the description in the model, combines data analysis and the automation of decision-making or operational processes. In the context of the solution, it is a component supporting the processing of input data, the application of analytical rules or models and the subsequent execution of automated responses through integrated tools and interfaces. This element is functionally linked to the solution's objective of using elements of automation and machine learning methods in the detection and evaluation of security events.

From the perspective of the overall architecture of the solution, the application layer creates a bridge between the business layer and the technological layer. On the one hand, it builds on the defined services and use cases; on the other hand, it creates the basis for the subsequent technological design of specific platforms and tools. Its significance lies in the fact that it names the main application responsibilities of the solution and creates a structured framework for the implementation, integration and further development of the individual functional areas of the SOCaaS platform.

1.1. Application Viewpoints

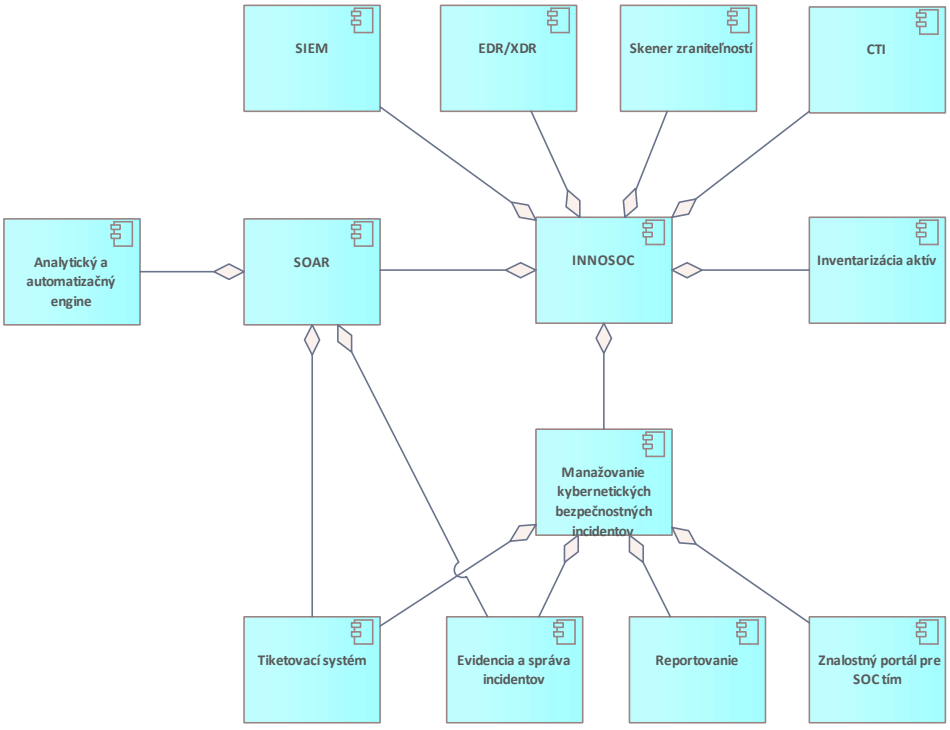


Figure 1: Application Viewpoints Components

The diagram presents a HL list of the application components of the solution.

Description of the elements used:

Analytics and automation engine

The analytics and automation engine is a software component or service that combines data analysis and the automation of decision-making or operational processes with the aim of increasing the efficiency, accuracy and speed of the system's response. In the context of cybersecurity or IT operations, this engine processes input data (e.g. logs, events, metrics), applies analytical rules or models (e.g. anomaly detection, correlation), and subsequently performs automated responses or actions through integrated tools and API interfaces.

CTI

CTI (Cyber Threat Intelligence) is a key building block of the SOCaaS architecture, as it provides the ability to enrich detection rules, correlation mechanisms and playbooks with external threat information – including IOCs, TTPs, attacker profiling, campaigns and related vulnerabilities. In the multi-tenant SOCaaS service environment, it plays a role in:

- Distribution and management of threat feeds across tenants,
- Automatic enrichment of SIEM and EDR data with known threats,
- Creation of attack timelines and mapping to MITRE ATT&CK techniques,
- Supporting threat hunting and forensic analysis processes,
- Prioritization of incidents according to the threat context.

For the solution design, open-source platforms were chosen that cover various aspects of CTI – from IOC collection through visualization to STIX/TAXII compatibility.

EDR/XDR

EDR (Endpoint Detection and Response) and XDR (Extended Detection and Response) are the fundamental technological pillars for threat detection at the endpoint level. In the SOCaaS environment, they serve for the collection and correlation of, and response to, events that would otherwise escape traditional perimeter detections. XDR extends the EDR concept with the ability to integrate data from networks, clouds, e-mail systems and other layers.

Incident recording and management

Incident recording and management is a core element of every security operation (SOC). This subcomponent ensures the systematic documentation, tracking and management of security events and incidents throughout their life cycle – from detection through analysis to closure. A good-quality recording system must enable teams to collaborate in a coordinated manner, document steps, record related artefacts (observables) and link incidents with relevant threats, systems or historical cases. Ideally, it also supports advanced data enrichment, auditing, multi-tenancy and the possibility of retrospective investigation of incidents.

INNOSOC

Represents a complex aggregated component.

Asset inventory

Asset inventory is a fundamental pillar of every security solution. It enables the SOC team to obtain an overview of all devices, system entities, applications and virtual or cloud resources that make up the organization's infrastructure. In the SOCaaS environment, it plays an even more significant role, as it enables centralized management, categorization, monitoring and tracking of the security status of the infrastructure across different customers.

Management of cybersecurity incidents

Effective handling of cyber incidents is a key pillar of the SOCaaS architecture. It involves not only the mere recording of events, but a comprehensive cycle of their recording, management, response, reporting and knowledge retention. For the SOC team it is essential that incidents are managed systematically, with an emphasis on traceability, analytical work, coordination and knowledge support. An effective and coordinated response to security incidents is an indispensable pillar of every SOCaaS solution. Within the INNOSOC architecture, four subcomponents were identified that together form an integrated system for the management, recording, documentation and handling of incidents:

1. Ticketing system
2. Incident recording and management
3. Reporting
4. Knowledge portal for the SOC team

Reporting

The reporting component within the SOCaaS architecture serves as the central means for aggregating, visualizing and distributing security information. Its role is to provide both regular and ad hoc outputs on the state of cybersecurity for various target groups – from SOC analysts, through team leaders, to the management of the organization or clients. Such a reporting system must cover:

- Generation of metrics on incidents, alerts, SLAs and KPIs,
- Overviews of trends over time and evaluation of the performance of security measures,
- Adaptation of content according to the level of detail (technical vs. strategic),
- Automation and scheduling of the distribution of outputs,
- Retrospective analysis and archiving of historical data.

SIEM

SIEM is the heart of the SOCaaS solution. It provides aggregation, processing, correlation and analysis of security events from various sources in real time. It enables analysts to identify anomalies, generate alerts and escalate incidents, while increasing visibility over the client's infrastructure.

Vulnerability scanner

The vulnerability scanner is a fundamental building block for ensuring a proactive approach in the SOCaaS architecture. It makes it possible to regularly identify weaknesses in the infrastructure, assess the status of updates, configurations and settings, and provide information for further prioritization and planning of corrective measures. The scanner outputs are often a source for SOAR systems, SLA metrics and reporting, as well as for assessing the client's security level.

SOAR

SOAR (Security Orchestration, Automation and Response) is the component responsible for automating incident responses, orchestrating security tools and supporting analysts' decision-making. In SOCaaS, it is a fundamental element enabling the scaling of response capacities without the need to increase staff numbers. It is indispensable especially when processing large volumes of alerts, where it reduces the MTTR and MTTC metrics.

Ticketing system

The ticketing system serves as the central communication and organizational channel for the management of security incidents throughout their entire life cycle. It enables the creation, assignment and tracking of tasks and communication between analysts, while supporting audit, SLA monitoring, escalation and notifications.

Knowledge portal for the SOC team

A knowledge portal for the SOC team is an important building block of every modern security operations centre. It serves as a central repository of know-how, playbooks, procedures, recommendations, analytical guides and often also historical lessons learned from incidents. The knowledge portal:

- Increases the efficiency and consistency of incident response,
- Enables rapid onboarding of new team members,
- Supports the standardization of processes and methodologies,
- Enables the team to retain and share proven procedures and its own analytical experience,
- Strengthens the ability to learn collectively from incidents and to build internal "threat intel" knowledge.

2. TECHNOLOGICAL ARCHITECTURE

The technological architecture represents a structured design of technological components, their mutual relationships and the manner of their integration into the overall ecosystem of the organization. It includes the definition of software layers, communication protocols, security mechanisms and operational processes. Its aim is to ensure optimal functioning of the system, its scalability, interoperability and ability to respond to changing business and regulatory requirements.

The project responds to the needs mainly of small and medium-sized organizations, which often lack sufficient expert capacity or budget to build their own security centre. The proposed solution provides centralized administration, automated incident detection and response, monitoring and reporting, thereby reducing the risk of cyberattacks and increasing the level of compliance.

1.2. Design of the technological architecture

The technological architecture determines the manner of deployment, operation and interconnection of the individual technical components, including containerized services, data storage, network layers and orchestration tools. It is designed with an emphasis on modularity, scalability, availability and security, while taking into account the requirements placed on a multi-tenant operating environment.

This section elaborates on specific technological elements – such as Kubernetes clusters, database services, CI/CD tools, container registries, proxy layers and security mechanisms – and their relationships to application components. The subsection also provides context for further technical specifications of the solution, including architecture viewpoints that map functional units such as CTI, EDR, SOAR, asset inventory, as well as data collection and analysis. The design of the technological architecture thus creates a reliable and well-manageable foundation for implementing the entire solution in line with the defined business objectives.

The proposed operation is planned on two identical environments differing only in the amount and type of resources:

- **PROD** environment – production environment with real data and operation of the public portal.
- **TEST** environment – testing, training, pre-production environment. The entire TEST environment is accessible via VPN for privileged users.

1.1.1. Technology Viewpoint CTI

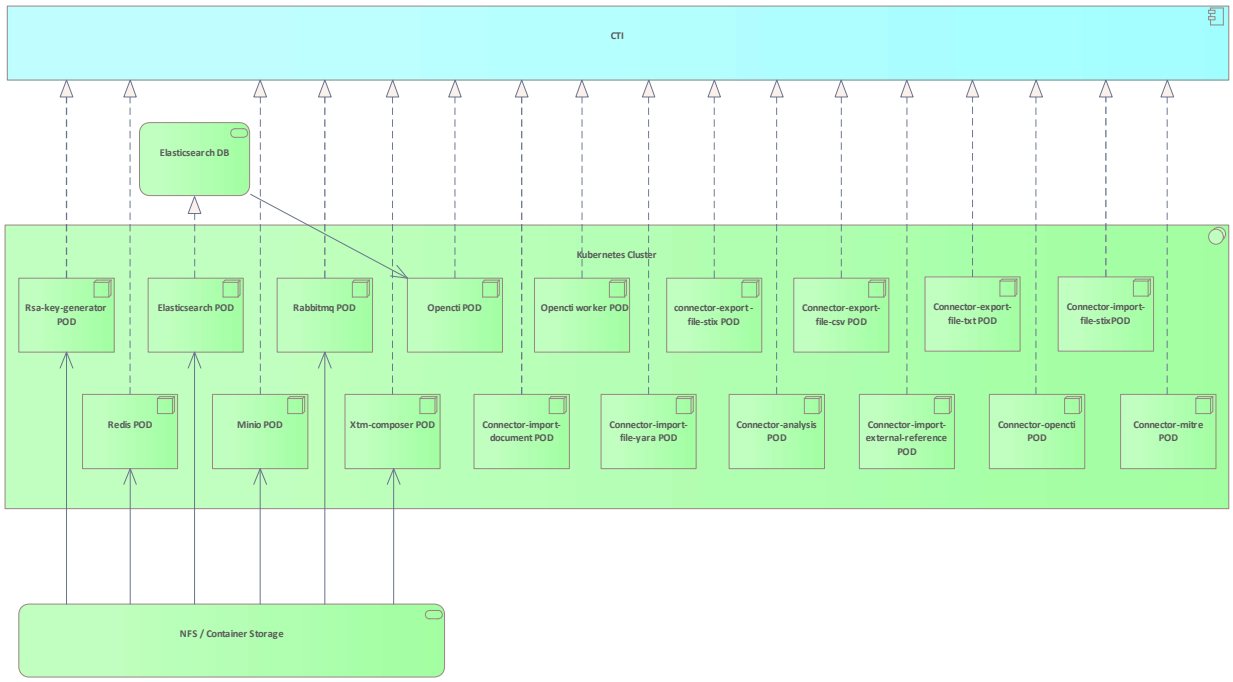


Figure 2: Technology Usage Viewpoint CTI alt 1

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 1 represents the use of the Elasticsearch database directly within the module setup. The design envisages a containerized version of NoSQL databases for Elasticsearch.

Description of the elements used:

CTI

CTI (Cyber Threat Intelligence) is a key building block of the SOCaaS architecture, as it provides the ability to enrich detection rules, correlation mechanisms and playbooks with external threat information – including IOCs, TTPs, attacker profiling, campaigns and related vulnerabilities. In the multi-tenant SOCaaS service environment, it plays a role in:

- Distribution and management of threat feeds across tenants,
- Automatic enrichment of SIEM and EDR data with known threats,
- Creation of attack timelines and mapping to MITRE ATT&CK techniques,
- Supporting threat hunting and forensic analysis processes,
- Prioritization of incidents according to the threat context.

For the solution design, open-source platforms were chosen that cover various aspects of CTI – from IOC collection through visualization to STIX/TAXII compatibility.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Elasticsearch DB

The technology service of the NoSQL Elasticsearch database is a high-performance, distributed system designed for indexing, searching and analysing data in real time, based on a document-oriented model. It uses a JSON document schema, supports horizontal scaling and high availability, and is optimized for processing large volumes of unstructured or semi-structured data such as logs, metrics or application records.

Elasticsearch POD

The Elasticsearch POD for the NoSQL database is a container instance deployed in a Kubernetes environment that hosts an Elasticsearch node as part of a distributed NoSQL database cluster. This POD provides indexing, searching and storage of document-oriented data, while enabling horizontal scaling, high availability and efficient processing of large volumes of unstructured information in real time.

Connector-analysis POD

The connector-analysis POD is a container unit deployed in a Kubernetes cluster that hosts an analysis connector for OpenCTI, intended for automated processing, analysis and enrichment of threat intelligence. This POD receives input data from OpenCTI, performs analytical operations on it (e.g. reputation checks, IoC processing or entity correlation) and returns the results back to the system for further intelligence activities and decision-making.

Connector-export-file-csv POD

The connector-export-file-csv POD is a container unit deployed in a Kubernetes cluster that hosts an export connector for OpenCTI, intended for generating and storing data in CSV (Comma-Separated Values) format. This POD enables automated export of threat intelligence, such as threat indicators, attack techniques or actors, to CSV files, thereby supporting their easy integration with external systems, reporting or manual analysis.

Connector-export-file-stix POD

The connector-export-file-stix POD is a container unit deployed in a Kubernetes cluster that hosts a connector for OpenCTI intended for exporting data to STIX (Structured Threat Information Expression) format. This POD enables automated generation and storage of threat information in STIX 2.1 format, thereby supporting standardized sharing of cybersecurity data with other systems, partners or communities.

Connector-export-file-txt POD

The connector-export-file-txt POD is a container unit deployed in a Kubernetes cluster that hosts an export connector for OpenCTI, intended for generating text files (TXT) containing selected threat information. This POD enables automated export of indicators of compromise (IoC) and other CTI data in the form of simple, readable text output for further use in external systems, tools or manual processes.

Connector-import-document POD

The connector-import-document POD is a container unit deployed in a Kubernetes cluster that hosts a connector for OpenCTI intended for automated import of documents containing intelligence or security information. This POD processes files in formats such as PDF, DOCX or TXT, extracts relevant data from them and subsequently imports them into the OpenCTI platform as structured objects for further analysis and correlation.

Connector-import-external-reference POD

The connector-import-external-reference POD is a container unit deployed in a Kubernetes cluster that hosts a connector for OpenCTI focused on importing external references from public or private sources of threat information. This POD regularly loads metadata and links to reports, articles or analytical documents, converts them into structured CTI (Cyber Threat Intelligence) objects and integrates them into OpenCTI for the purposes of correlation and analytical enrichment of the threat profile.

Connector-import-file-stixPOD

The connector-import-file-stix POD is a container unit deployed in a Kubernetes cluster that hosts an import connector for OpenCTI, focused on loading and processing files in STIX (Structured Threat Information Expression) format. This POD automatically imports threat information from STIX 2.0/2.1 files into OpenCTI, where it is subsequently transformed into structured CTI objects and integrated into existing data for the purposes of correlation, visualization and intelligence analysis.

Connector-import-file-yara POD

The connector-import-file-yara POD is a container unit deployed in a Kubernetes cluster that hosts a connector for OpenCTI focused on importing YARA files – rules used for detecting malware and threats. This POD automatically loads YARA files from a defined repository, analyses their content and converts them into structured threat objects, which it then imports into OpenCTI for the purposes of managing and correlating intelligence.

Connector-mitre POD

The connector-mitre POD is a container unit deployed in a Kubernetes cluster that hosts a connector for OpenCTI intended for automated import of data from the MITRE ATT&CK framework. This POD regularly downloads and synchronizes the official tactics, techniques, procedures (TTPs), groups, software and campaigns from the MITRE database and subsequently transforms them into structured CTI objects in OpenCTI, thereby supporting accurate and up-to-date analysis of cyber threats.

Connector-opencti POD

The connector-opencti POD is a container unit deployed in a Kubernetes cluster that hosts the main OpenCTI connector, providing bidirectional communication between the OpenCTI platform and the individual connectors. This POD serves as middleware that registers connectors, coordinates their activity and ensures data exchange via the OpenCTI API and a message broker (e.g. RabbitMQ), thereby enabling scalable, modular and efficient processing of cybersecurity information.

Minio POD

The MinIO POD is a container unit deployed in a Kubernetes cluster that hosts the MinIO service – a high-performance, scalable and S3-compatible object storage system. This POD provides storage for unstructured data such as logs, backups, multimedia files or outputs of analytical tools, while supporting high availability, replication and integration with modern cloud-native applications.

Opencti POD

The OpenCTI POD is a container unit deployed in a Kubernetes cluster that hosts the OpenCTI (Open Cyber Threat Intelligence) platform – an open-source solution for managing, analysing and correlating cyber threats and intelligence (Threat Intelligence). This POD provides a central hub for collecting, visualizing and linking threat information, including entities such as TTPs (tactics, techniques, procedures), indicators of compromise (IoC) or threat actors, and supports integration with other security tools via an API.

Opencti worker POD

The OpenCTI Worker POD is a container unit deployed in a Kubernetes cluster that performs asynchronous processing of tasks and data operations within the OpenCTI platform. This POD provides indexing, transformation, correlation and enrichment of cybersecurity information, while working closely with the main OpenCTI server and the connectors to ensure smooth data integration and computational distribution.

Rabbitmq POD

The RabbitMQ POD is a container unit deployed in a Kubernetes cluster that hosts a RabbitMQ server – a reliable and widely used message broker implementing protocols such as AMQP, MQTT or STOMP. This POD provides asynchronous communication and message exchange between services and system components, thereby supporting scalability, resilience and efficient task processing in distributed applications.

Redis POD

The redis POD is a container unit deployed in a Kubernetes cluster that hosts a Redis server – a high-performance in-memory NoSQL key-value database. This POD is used for fast storage, reading and manipulation of temporary or persistent data, often used for caching, message queuing, session storage and sharing state between microservices.

Rsa-key-generator POD

The rsa-key-generator POD is a container unit deployed in a Kubernetes cluster that is used to generate RSA key pairs for the purposes of encryption, digital signing or secure communication. This POD provides automated and secure creation of cryptographic keys, which can subsequently be used by other services within the infrastructure, for example for TLS certificates, GPG signatures or authentication.

Xtm-composer POD

The xtm-composer POD is a container unit deployed in a Kubernetes cluster that hosts the XTM Composer service – a component intended for composing and generating data or security outputs within a larger system, often related to cybersecurity management or automated reporting. This POD provides processing of input data, its transformation and output in a predefined format, and may cooperate with other components such as databases, analytical tools or API gateways.

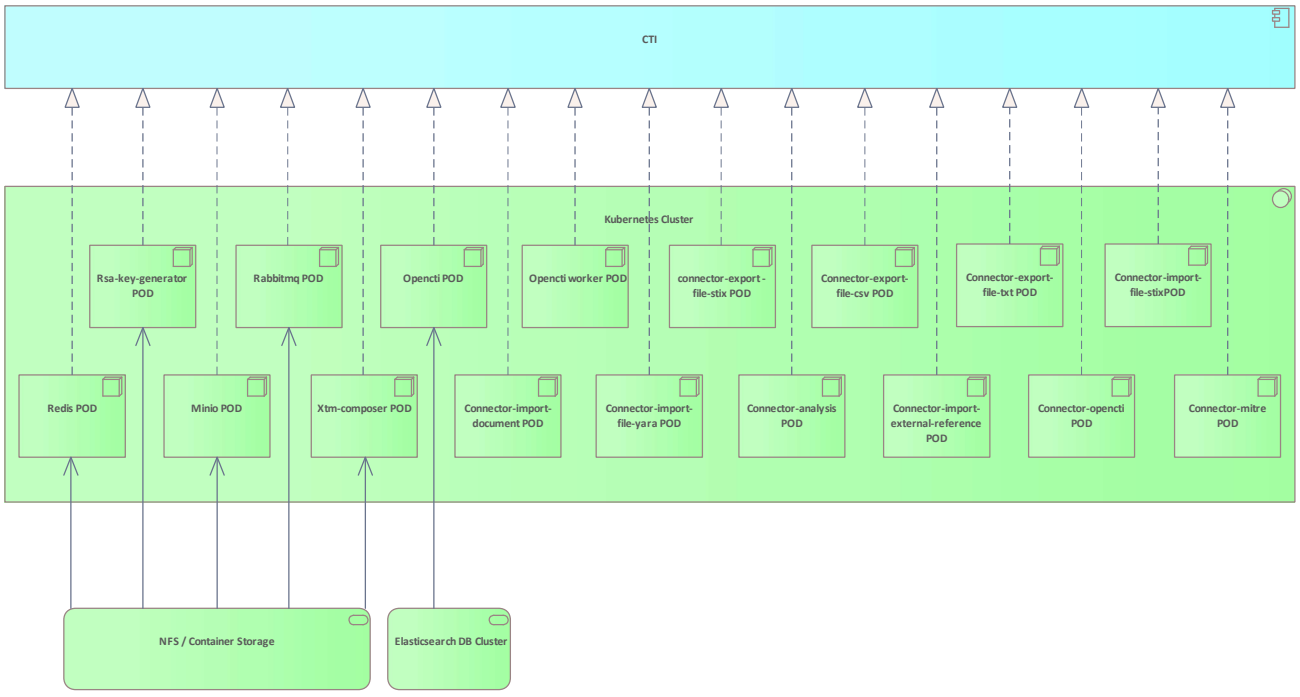


Figure 3: Technology Usage Viewpoint CTI alt 2

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 2 represents the use of the Elasticsearch database directly in the form of a centralized DB Cluster.

Description of the elements used:

CTI

CTI (Cyber Threat Intelligence) is a key building block of the SOCaaS architecture, as it provides the ability to enrich detection rules, correlation mechanisms and playbooks with external threat information – including IOCs, TTPs, attacker profiling, campaigns and related vulnerabilities. In the multi-tenant SOCaaS service environment, it plays a role in:

- Distribution and management of threat feeds across tenants,
- Automatic enrichment of SIEM and EDR data with known threats,
- Creation of attack timelines and mapping to MITRE ATT&CK techniques,
- Supporting threat hunting and forensic analysis processes,
- Prioritization of incidents according to the threat context.

For the solution design, open-source platforms were chosen that cover various aspects of CTI – from IOC collection through visualization to STIX/TAXII compatibility.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Elasticsearch DB Cluster

The technology service of the NoSQL Elasticsearch database in the form of a cluster represents a multi-node distributed solution in which the individual nodes cooperate on storing, indexing and searching document-oriented data. Each node in the cluster performs a specific role (e.g. master, data or ingest node), which enables horizontal scaling, high availability and fault tolerance, while the system as a whole ensures processing of large volumes of data in real time.

Connector-analysis POD

The connector-analysis POD is a container unit deployed in a Kubernetes cluster that hosts an analysis connector for OpenCTI, intended for automated processing, analysis and enrichment of threat intelligence. This POD receives input data from OpenCTI, performs analytical operations on it (e.g. reputation checks, IoC processing or entity correlation) and returns the results back to the system for further intelligence activities and decision-making.

Connector-export-file-csv POD

The connector-export-file-csv POD is a container unit deployed in a Kubernetes cluster that hosts an export connector for OpenCTI, intended for generating and storing data in CSV (Comma-Separated Values) format. This POD enables automated export of threat intelligence, such as threat indicators, attack techniques or actors, to CSV files, thereby supporting their easy integration with external systems, reporting or manual analysis.

Connector-export-file-stix POD

The connector-export-file-stix POD is a container unit deployed in a Kubernetes cluster that hosts a connector for OpenCTI intended for exporting data to STIX (Structured Threat Information Expression) format. This POD enables automated generation and storage of threat information in STIX 2.1 format, thereby supporting standardized sharing of cybersecurity data with other systems, partners or communities.

Connector-export-file-txt POD

The connector-export-file-txt POD is a container unit deployed in a Kubernetes cluster that hosts an export connector for OpenCTI, intended for generating text files (TXT) containing selected threat information. This POD enables automated export of indicators of compromise (IoC) and other CTI data in the form of simple, readable text output for further use in external systems, tools or manual processes.

Connector-import-document POD

The connector-import-document POD is a container unit deployed in a Kubernetes cluster that hosts a connector for OpenCTI intended for automated import of documents containing intelligence or security information. This POD processes files in formats such as PDF, DOCX or TXT, extracts relevant data from them and subsequently imports them into the OpenCTI platform as structured objects for further analysis and correlation.

Connector-import-external-reference POD

The connector-import-external-reference POD is a container unit deployed in a Kubernetes cluster that hosts a connector for OpenCTI focused on importing external references from public or private sources of threat information. This POD regularly loads metadata and links to reports, articles or analytical documents, converts them into structured CTI (Cyber Threat Intelligence) objects and integrates them into OpenCTI for the purposes of correlation and analytical enrichment of the threat profile.

Connector-import-file-stixPOD

The connector-import-file-stix POD is a container unit deployed in a Kubernetes cluster that hosts an import connector for OpenCTI, focused on loading and processing files in STIX (Structured Threat Information Expression) format. This POD automatically imports threat information from STIX 2.0/2.1 files into OpenCTI, where it is subsequently transformed into structured CTI objects and integrated into existing data for the purposes of correlation, visualization and intelligence analysis.

Connector-import-file-yara POD

The connector-import-file-yara POD is a container unit deployed in a Kubernetes cluster that hosts a connector for OpenCTI focused on importing YARA files – rules used for detecting malware and threats. This POD automatically loads YARA files from a defined repository, analyses their content and converts them into structured threat objects, which it then imports into OpenCTI for the purposes of managing and correlating intelligence.

Connector-mitre POD

The connector-mitre POD is a container unit deployed in a Kubernetes cluster that hosts a connector for OpenCTI intended for automated import of data from the MITRE ATT&CK framework. This POD regularly downloads and synchronizes the official tactics, techniques, procedures (TTPs), groups, software and campaigns from the MITRE database and subsequently transforms them into structured CTI objects in OpenCTI, thereby supporting accurate and up-to-date analysis of cyber threats.

Connector-opencti POD

The connector-opencti POD is a container unit deployed in a Kubernetes cluster that hosts the main OpenCTI connector, providing bidirectional communication between the OpenCTI platform and the individual connectors. This POD serves as middleware that registers connectors, coordinates their activity and ensures data exchange via the OpenCTI API and a message broker (e.g. RabbitMQ), thereby enabling scalable, modular and efficient processing of cybersecurity information.

Minio POD

The MinIO POD is a container unit deployed in a Kubernetes cluster that hosts the MinIO service – a high-performance, scalable and S3-compatible object storage system. This POD provides storage for unstructured data such as logs, backups, multimedia files or outputs of analytical tools, while supporting high availability, replication and integration with modern cloud-native applications.

Opencti POD

The OpenCTI POD is a container unit deployed in a Kubernetes cluster that hosts the OpenCTI (Open Cyber Threat Intelligence) platform – an open-source solution for managing, analysing and correlating cyber threats and intelligence (Threat Intelligence). This POD provides a central hub for collecting, visualizing and linking threat information, including entities such as TTPs (tactics, techniques, procedures), indicators of compromise (IoC) or threat actors, and supports integration with other security tools via an API.

Opencti worker POD

The OpenCTI Worker POD is a container unit deployed in a Kubernetes cluster that performs asynchronous processing of tasks and data operations within the OpenCTI platform. This POD provides indexing, transformation, correlation and enrichment of cybersecurity information, while working closely with the main OpenCTI server and the connectors to ensure smooth data integration and computational distribution.

Rabbitmq POD

The RabbitMQ POD is a container unit deployed in a Kubernetes cluster that hosts a RabbitMQ server – a reliable and widely used message broker implementing protocols such as AMQP, MQTT or STOMP. This POD provides asynchronous communication and message exchange between services and system components, thereby supporting scalability, resilience and efficient task processing in distributed applications.

Redis POD

The redis POD is a container unit deployed in a Kubernetes cluster that hosts a Redis server – a high-performance in-memory NoSQL key-value database. This POD is used for fast storage, reading and manipulation of temporary or persistent data, often used for caching, message queuing, session storage and sharing state between microservices.

Rsa-key-generator POD

The rsa-key-generator POD is a container unit deployed in a Kubernetes cluster that is used to generate RSA key pairs for the purposes of encryption, digital signing or secure communication. This POD provides automated and secure creation of cryptographic keys, which can subsequently be used by other services within the infrastructure, for example for TLS certificates, GPG signatures or authentication.

Xtm-composer POD

The xtm-composer POD is a container unit deployed in a Kubernetes cluster that hosts the XTM Composer service – a component intended for composing and generating data or security outputs within a larger system, often related to cybersecurity management or automated reporting. This POD provides processing of input data, its transformation and output in a predefined format, and may cooperate with other components such as databases, analytical tools or API gateways.

1.1.2. Technology Viewpoint EDR/XDR

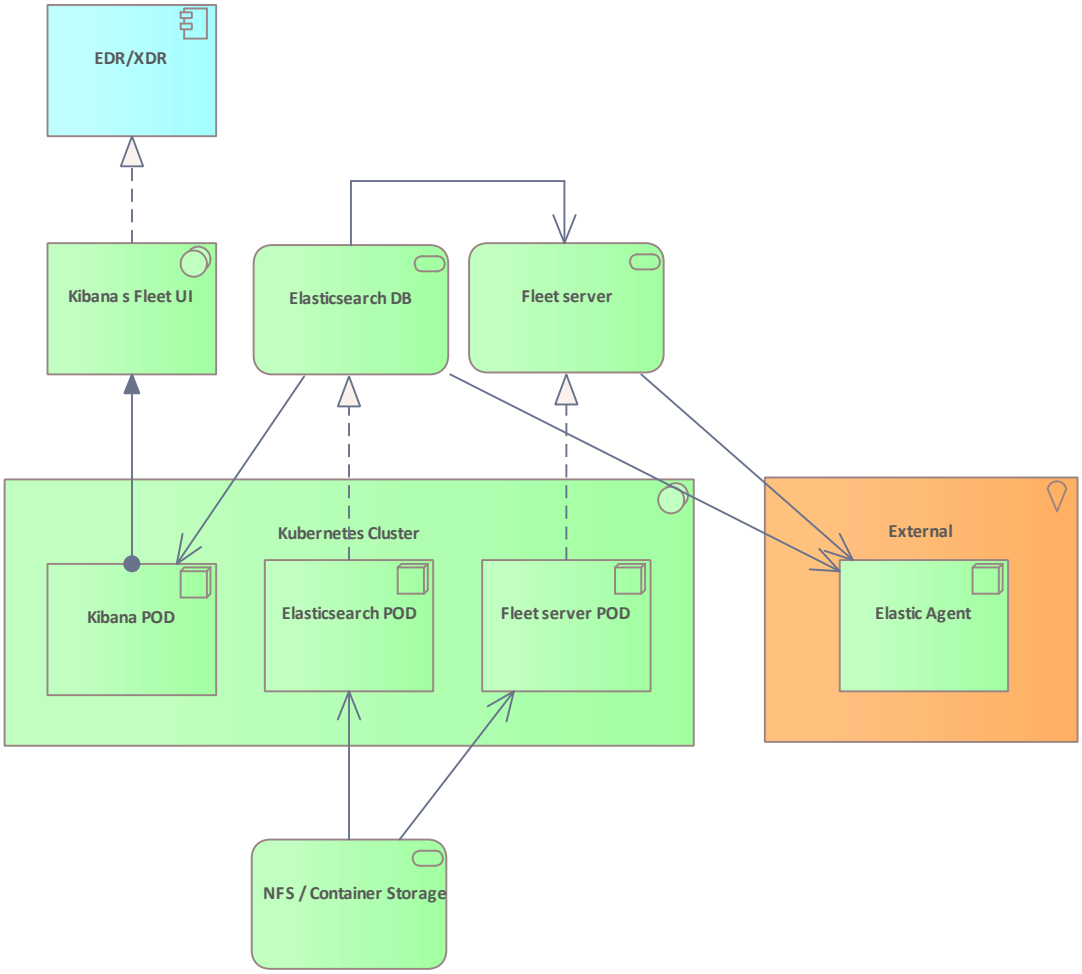


Figure 4: Technology Usage Viewpoint EDR/XDR alt 1

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 1 represents the use of the Elasticsearch database directly within the module setup. The design envisages a containerized version of NoSQL databases for Elasticsearch.

Description of the elements used:

EDR/XDR

EDR (Endpoint Detection and Response) and XDR (Extended Detection and Response) are the fundamental technological pillars for threat detection at the endpoint level. In the SOCaaS environment, they serve for the collection and correlation of, and response to, events that would otherwise escape traditional perimeter detections. XDR extends the EDR concept with the ability to integrate data from networks, clouds, e-mail systems and other layers.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Elasticsearch DB

The technology service of the NoSQL Elasticsearch database is a high-performance, distributed system designed for indexing, searching and analysing data in real time, based on a document-oriented model. It uses a JSON document schema, supports horizontal scaling and high availability, and is optimized for processing large volumes of unstructured or semi-structured data such as logs, metrics or application records.

Elasticsearch POD

The Elasticsearch POD for the NoSQL database is a container instance deployed in a Kubernetes environment that hosts an Elasticsearch node as part of a distributed NoSQL database cluster. This POD provides indexing, searching and storage of document-oriented data, while enabling horizontal scaling, high availability and efficient processing of large volumes of unstructured information in real time.

External

Represents the physical location at the client and thus outside the INNOSOC environment.

Kibana POD

The Kibana POD for SIEM is a container instance deployed in a Kubernetes cluster that provides a graphical user interface for visualizing and analysing security data collected in Elasticsearch. It serves as a frontend component for the SIEM (Security Information and Event Management) solution, enabling interactive monitoring, event correlation, creation of dashboards and detection of security incidents in real time.

Fleet server

The Fleet Server technology service is a central component of the Elastic Stack used for the management, coordination and secure communication of Elastic Agents deployed on various systems. Fleet Server relays configurations, updates and commands from the Kibana Fleet UI to the individual agents, thereby enabling centralized management of the collection of logs, metrics and security data within large distributed environments.

Elastic Agent

Represents an Elastic Agent instance for monitoring and processing logs and writing them to Elasticsearch, and for protecting the host against security threats.

Fleet server POD

The Elasticsearch POD for the NoSQL database is a container unit deployed in a Kubernetes cluster that hosts an Elasticsearch node as part of a distributed search and indexing system. This POD provides storage, searching and analysis of document-oriented data in real time, with support for horizontal scaling, high availability and integration with other tools in a data-analytics or security ecosystem.

Kibana with Fleet UI

Kibana with Fleet UI is an extended web interface of the Kibana tool that provides centralized management of agents and data integrations within the Elastic Stack. Fleet UI enables simple deployment, configuration and monitoring of Elastic Agents, thereby simplifying the collection of logs, metrics and security data from various systems and their subsequent visualization and analysis in Kibana.

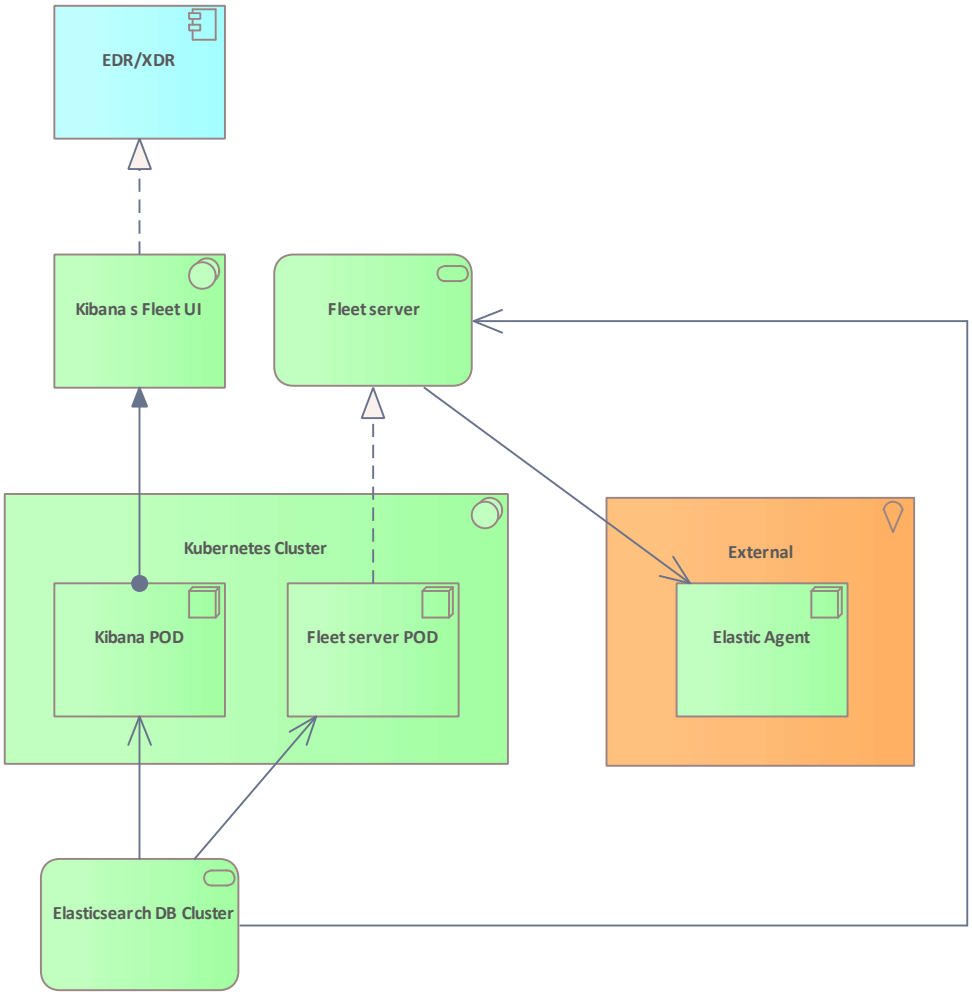


Figure 5: Technology Usage Viewpoint EDR/XDR alt 2

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 2 represents the use of the Elasticsearch database directly in the form of a centralized DB Cluster.

Description of the elements used:

EDR/XDR

EDR (Endpoint Detection and Response) and XDR (Extended Detection and Response) are the fundamental technological pillars for threat detection at the endpoint level. In the SOCaaS environment, they serve for the collection and correlation of, and response to, events that would otherwise escape traditional perimeter detections. XDR extends the EDR concept with the ability to integrate data from networks, clouds, e-mail systems and other layers.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment,

scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Elasticsearch DB Cluster

The technology service of the NoSQL Elasticsearch database in the form of a cluster represents a multi-node distributed solution in which the individual nodes cooperate on storing, indexing and searching document-oriented data. Each node in the cluster performs a specific role (e.g. master, data or ingest node), which enables horizontal scaling, high availability and fault tolerance, while the system as a whole ensures processing of large volumes of data in real time.

External

Represents the physical location at the client and thus outside the INNOSOC environment.

Kibana POD

The Kibana POD for SIEM is a container instance deployed in a Kubernetes cluster that provides a graphical user interface for visualizing and analysing security data collected in Elasticsearch. It serves as a frontend component for the SIEM (Security Information and Event Management) solution, enabling interactive monitoring, event correlation, creation of dashboards and detection of security incidents in real time.

Fleet server

The Fleet Server technology service is a central component of the Elastic Stack used for the management, coordination and secure communication of Elastic Agents deployed on various systems. Fleet Server relays configurations, updates and commands from the Kibana Fleet UI to the individual agents, thereby enabling centralized management of the collection of logs, metrics and security data within large distributed environments.

Elastic Agent

Represents an Elastic Agent instance for monitoring and processing logs and writing them to Elasticsearch, and for protecting the host against security threats.

Fleet server POD

The Elasticsearch POD for the NoSQL database is a container unit deployed in a Kubernetes cluster that hosts an Elasticsearch node as part of a distributed search and indexing system. This POD provides storage, searching and analysis of document-oriented data in real time, with support for horizontal scaling, high availability and integration with other tools in a data-analytics or security ecosystem.

Kibana with Fleet UI

Kibana with Fleet UI is an extended web interface of the Kibana tool that provides centralized management of agents and data integrations within the Elastic Stack. Fleet UI enables simple deployment, configuration and monitoring of Elastic Agents, thereby simplifying the collection of logs, metrics and security data from various systems and their subsequent visualization and analysis in Kibana.

1.1.3. Technology Viewpoint INNOSOC Common

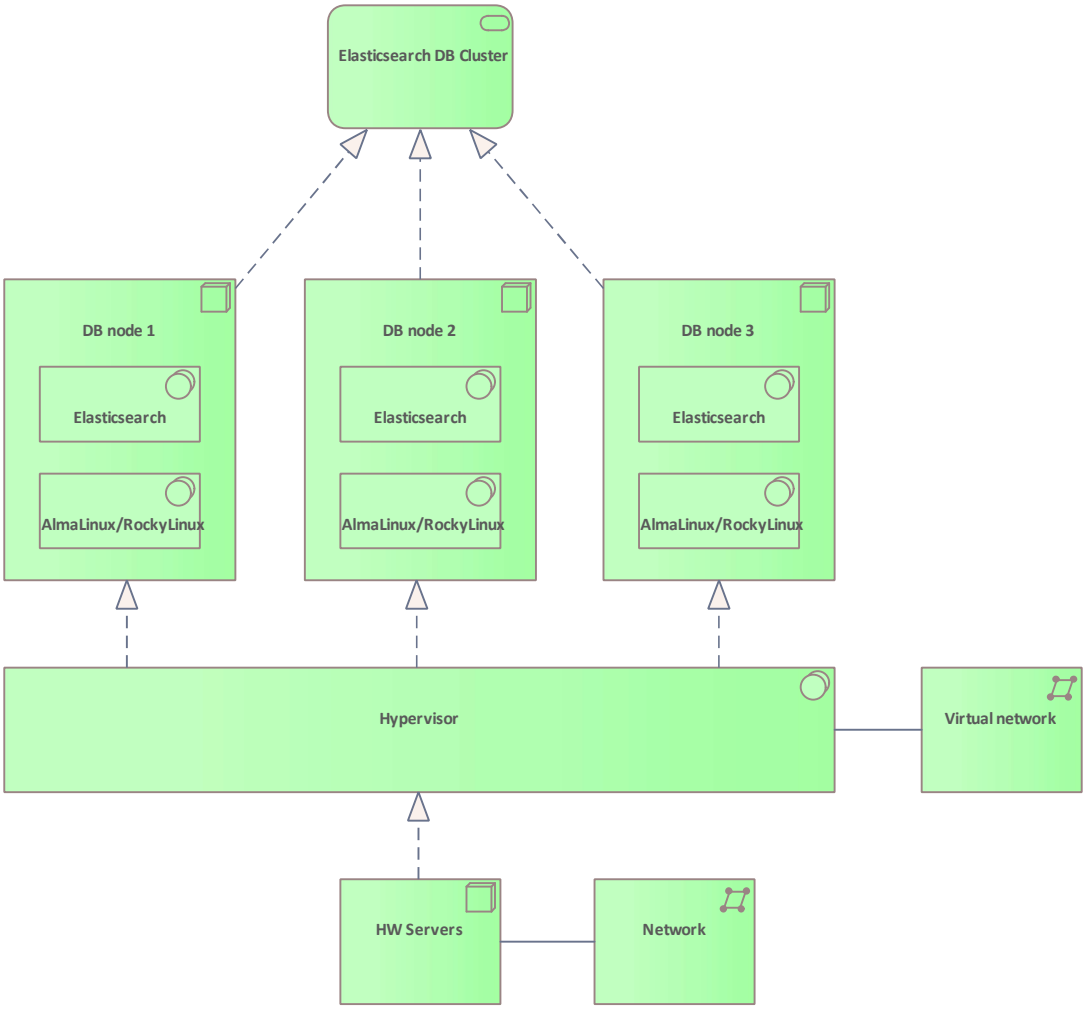


Figure 6 : Technology Viewpoint DB Cluster

The diagram shows the technological architecture of the PostgreSQL Cluster.

Description of the elements used:

DB node 1

Represents Node 1 of the Elasticsearch Cluster. An Elasticsearch Cluster node is an individual Elasticsearch instance that is part of a distributed cluster and performs one or more specific roles, such as data management, indexing or coordination of requests. Each node communicates with the other nodes in the cluster, and together they ensure scalability, high availability and parallel processing of queries and operations on large volumes of data.

DB node 2

Represents Node 2 of the Elasticsearch Cluster. An Elasticsearch Cluster node is an individual Elasticsearch instance that is part of a distributed cluster and performs one or more specific roles, such as data management, indexing or coordination of requests. Each node communicates with the other nodes in the cluster, and together they ensure scalability, high availability and parallel processing of queries and operations on large volumes of data.

DB node 3

Represents Node 3 of the Elasticsearch Cluster. An Elasticsearch Cluster node is an individual Elasticsearch instance that is part of a distributed cluster and performs one or more specific roles, such as data management, indexing or coordination of requests. Each node communicates with the other nodes in the cluster, and together they ensure scalability, high availability and parallel processing of queries and operations on large volumes of data.

HW Servers

Represents the HW server infrastructure. The HW server infrastructure represents the physical computing and network equipment that forms the basis for operating IT systems, including servers, storage, switches, routers and backup power supplies. This infrastructure is deployed either on-premise or as part of a cloud platform, and provides computing power, availability and connectivity for both virtualized and non-virtualized environments.

Hypervisor

Represents a Type 1 or Type 2 Hypervisor. A Type 1 hypervisor, also known as a "bare-metal" hypervisor, runs directly on the physical hardware without an operating system, thereby providing high performance and efficiency of virtualization; examples include VMware ESXi, Microsoft Hyper-V (core) or Xen. This type of hypervisor is intended mainly for data centres and enterprise environments where emphasis is placed on performance, stability and security.

A Type 2 hypervisor, also called a "hosted" hypervisor, runs on top of an existing operating system as an application, and thus shares system resources with the host; examples include VMware Workstation or Oracle VirtualBox. Although it is simpler to install and suitable for development or testing, its performance and isolation are lower compared with Type 1.

Network

Represents the network/subnet of the HW infrastructure. The network or subnet of the HW infrastructure is the physical and logical layer interconnecting servers, storage and network devices, which provides data transfer, communication and access to services within the data centre.

Virtual network

Represents the virtual network/subnet of the Hypervisor. The virtual network or subnet of the Hypervisor is a logical network infrastructure that the hypervisor creates for the purpose of interconnecting virtual machines (VMs) with one another, with the host system or with the external network. It enables the definition of virtual switches and NAT, bridged or internal

networking modes, thereby providing flexibility in the isolation, communication and routing of data between VMs and the physical network environment.

AlmaLinux/RockyLinux

Represents the AlmaLinux/RockyLinux Linux distribution. AlmaLinux and RockyLinux are open-source Linux distributions based on Red Hat Enterprise Linux (RHEL) that provide fully binary-compatible alternatives to RHEL without licensing restrictions. They are designed for stable, secure and long-term supported server environments, and are often used in enterprise, cloud and data centres as a replacement for CentOS after its discontinuation as a downstream project.

Elasticsearch

Represents the Elasticsearch NoSQL database. Elasticsearch is a high-performance, distributed NoSQL database designed for indexing, full-text searching and analysing large volumes of data in real time. It uses a JSON-based document-oriented model, supports horizontal scaling and high availability, and is widely used in areas such as log management, monitoring, security analytics and business intelligence.

AlmaLinux/RockyLinux

Represents the AlmaLinux/RockyLinux Linux distribution. AlmaLinux and RockyLinux are open-source Linux distributions based on Red Hat Enterprise Linux (RHEL) that provide fully binary-compatible alternatives to RHEL without licensing restrictions. They are designed for stable, secure and long-term supported server environments, and are often used in enterprise, cloud and data centres as a replacement for CentOS after its discontinuation as a downstream project.

Elasticsearch

Represents the Elasticsearch NoSQL database. Elasticsearch is a high-performance, distributed NoSQL database designed for indexing, full-text searching and analysing large volumes of data in real time. It uses a JSON-based document-oriented model, supports horizontal scaling and high availability, and is widely used in areas such as log management, monitoring, security analytics and business intelligence.

AlmaLinux/RockyLinux

Represents the AlmaLinux/RockyLinux Linux distribution. AlmaLinux and RockyLinux are open-source Linux distributions based on Red Hat Enterprise Linux (RHEL) that provide fully binary-compatible alternatives to RHEL without licensing restrictions. They are designed for stable, secure and long-term supported server environments, and are often used in enterprise, cloud and data centres as a replacement for CentOS after its discontinuation as a downstream project.

Elasticsearch

Represents the Elasticsearch NoSQL database. Elasticsearch is a high-performance, distributed NoSQL database designed for indexing, full-text searching and analysing large

volumes of data in real time. It uses a JSON-based document-oriented model, supports horizontal scaling and high availability, and is widely used in areas such as log management, monitoring, security analytics and business intelligence.

Elasticsearch DB Cluster

The technology service of the NoSQL Elasticsearch database in the form of a cluster represents a multi-node distributed solution in which the individual nodes cooperate on storing, indexing and searching document-oriented data. Each node in the cluster performs a specific role (e.g. master, data or ingest node), which enables horizontal scaling, high availability and fault tolerance, while the system as a whole ensures processing of large volumes of data in real time.

1.1.4. Technology Viewpoint Asset Inventory

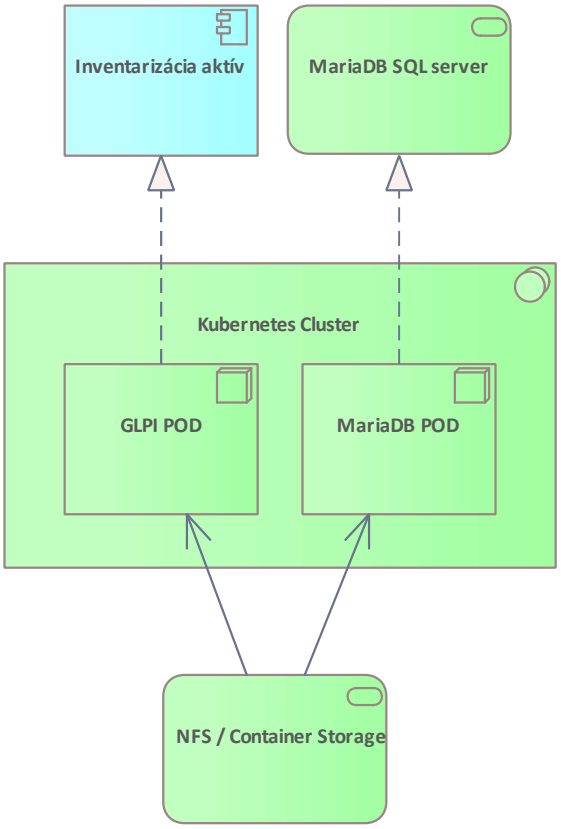


Figure 7: Technology Usage Viewpoint Asset Inventory alt 1

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 1 represents the use of the database directly within the module setup. The design envisages a containerized version of the SQL server.

Description of the elements used:

Asset inventory

Asset inventory is a fundamental pillar of every security solution. It enables the SOC team to obtain an overview of all devices, system entities, applications and virtual or cloud resources that make up the organization's infrastructure. In the SOCaaS environment, it plays an even more significant role, as it enables centralized management, categorization, monitoring and tracking of the security status of the infrastructure across different customers.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

MariaDB SQL server

Represents the technology service of the relational MariaDB SQL database. A technology service in the field of database systems refers to a software component or platform that provides a consistent, managed and scalable environment for storing, manipulating and managing structured data. In the case of the MariaDB relational database, it is a platform-independent database service that makes the relational model accessible using SQL (Structured Query Language) through a consistent and transactionally safe application interface.

GLPI POD

Represents the GLPI POD for asset inventory. GLPI (Gestionnaire Libre de Parc Informatique) is an open-source IT service management (ITSM) system that offers functionalities such as IT asset management (hardware, software), incident management, service request management, inventory and user management, and a CMDB (Configuration Management Database). A POD (Point Of Deployment) in the context of GLPI represents a modular and distributed deployment of GLPI components at multiple sites or locations, with each POD acting as a local node for the collection, updating and management of IT assets.

MariaDB POD

Represents the MariaDB POD for the relational SQL DB. MariaDB is an advanced open-source relational database management system (RDBMS) that originated as a fork of MySQL. It complies with the SQL (Structured Query Language) standard and provides powerful tools for managing and manipulating relational data. MariaDB supports the classic

ACID architecture (Atomicity, Consistency, Isolation, Durability) and MVCC (Multiversion Concurrency Control) for safe and consistent database transactions.

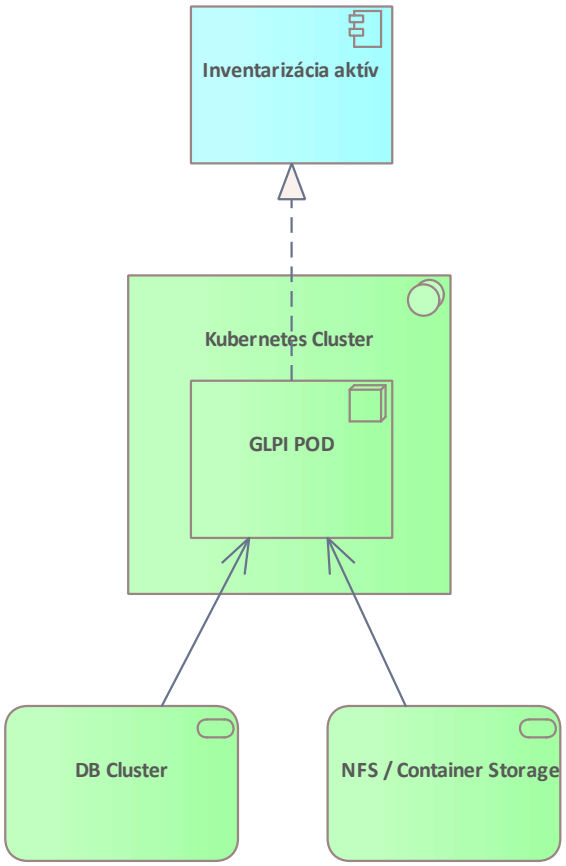


Figure 8: Technology Usage Viewpoint Asset Inventory alt 2

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 2 represents the use of the database directly in the form of a centralized DB Cluster.

Description of the elements used:

Asset inventory

Asset inventory is a fundamental pillar of every security solution. It enables the SOC team to obtain an overview of all devices, system entities, applications and virtual or cloud resources that make up the organization's infrastructure. In the SOCaaS environment, it plays an even more significant role, as it enables centralized management, categorization, monitoring and tracking of the security status of the infrastructure across different customers.

DB Cluster

Represents a clustered relational database service. A clustered relational database service is a distributed database system that provides relational database services through multiple interconnected database nodes (clusters), thereby ensuring high availability, scalability, redundancy and fault tolerance.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

GLPI POD

Represents the GLPI POD for asset inventory. GLPI (Gestionnaire Libre de Parc Informatique) is an open-source IT service management (ITSM) system that offers functionalities such as IT asset management (hardware, software), incident management, service request management, inventory and user management, and a CMDB (Configuration Management Database). A POD (Point Of Deployment) in the context of GLPI represents a modular and distributed deployment of GLPI components at multiple sites or locations, with each POD acting as a local node for the collection, updating and management of IT assets.

1.1.5. Technology Viewpoint Kubernetes

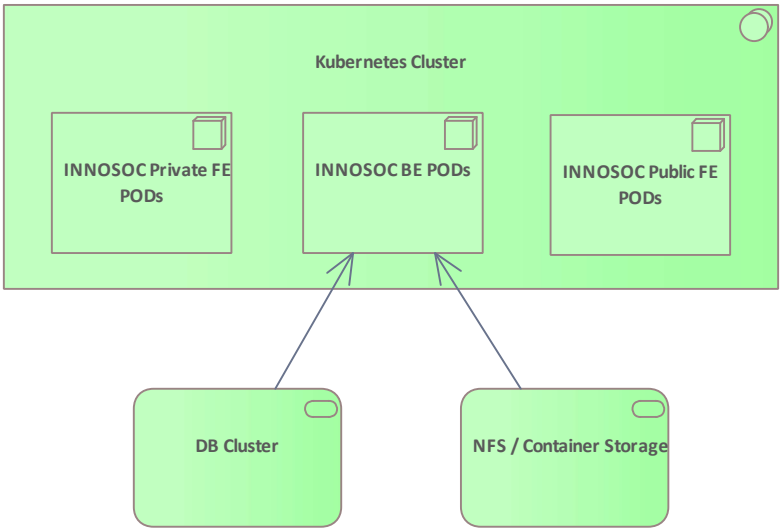


Figure 9: Technology Usage Viewpoint and Containers maping

The diagram shows the assignment of the individual PODs to the corresponding application components.

Description of the elements used:

DB Cluster

Represents a clustered relational database service. A clustered relational database service is a distributed database system that provides relational database services through multiple interconnected database nodes (clusters), thereby ensuring high availability, scalability, redundancy and fault tolerance.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

INNOSOC BE PODs

Represents the BE PODs. BE PODs refer to container instances of the system's backend services (BE – Backend) deployed in a distributed environment, typically in a Kubernetes cluster or other container orchestration. These PODs form the server (logical) layer of the architecture, which provides data processing, business logic, transactions, API services and interaction with the database and network entities (core network functions).

INNOSOC Private FE PODs

FE Private PODs represent isolated and modular deployment units of the system's frontend components in the customer's private environment. This is an architectural element based on the principles of containerization and microservice architecture, where each POD contains independently deployable elements of the presentation layer (FE – frontend), such as web applications, Single Page Applications (SPA) or UI modules.

INNOSOC Public FE PODs

FE Public PODs represent computing deployment units of the system's frontend components in a publicly accessible cloud or hybrid environment managed by the service provider or by the organization's centralized infrastructure. Within this architecture, the frontend components – typically Single Page Applications (SPA) or web interfaces – are operated as part of a multi-tenant system accessible via a public network (the internet), with communication and access governed by strict security policies. These PODs function as isolated deployments within a centralized cloud (e.g. Microsoft Azure, AWS, GCP) or in a managed Kubernetes cluster. Their aim is to provide a scalable, highly available and easily updatable interface for end users or customers who access the services without the need for a local installation or private infrastructure.

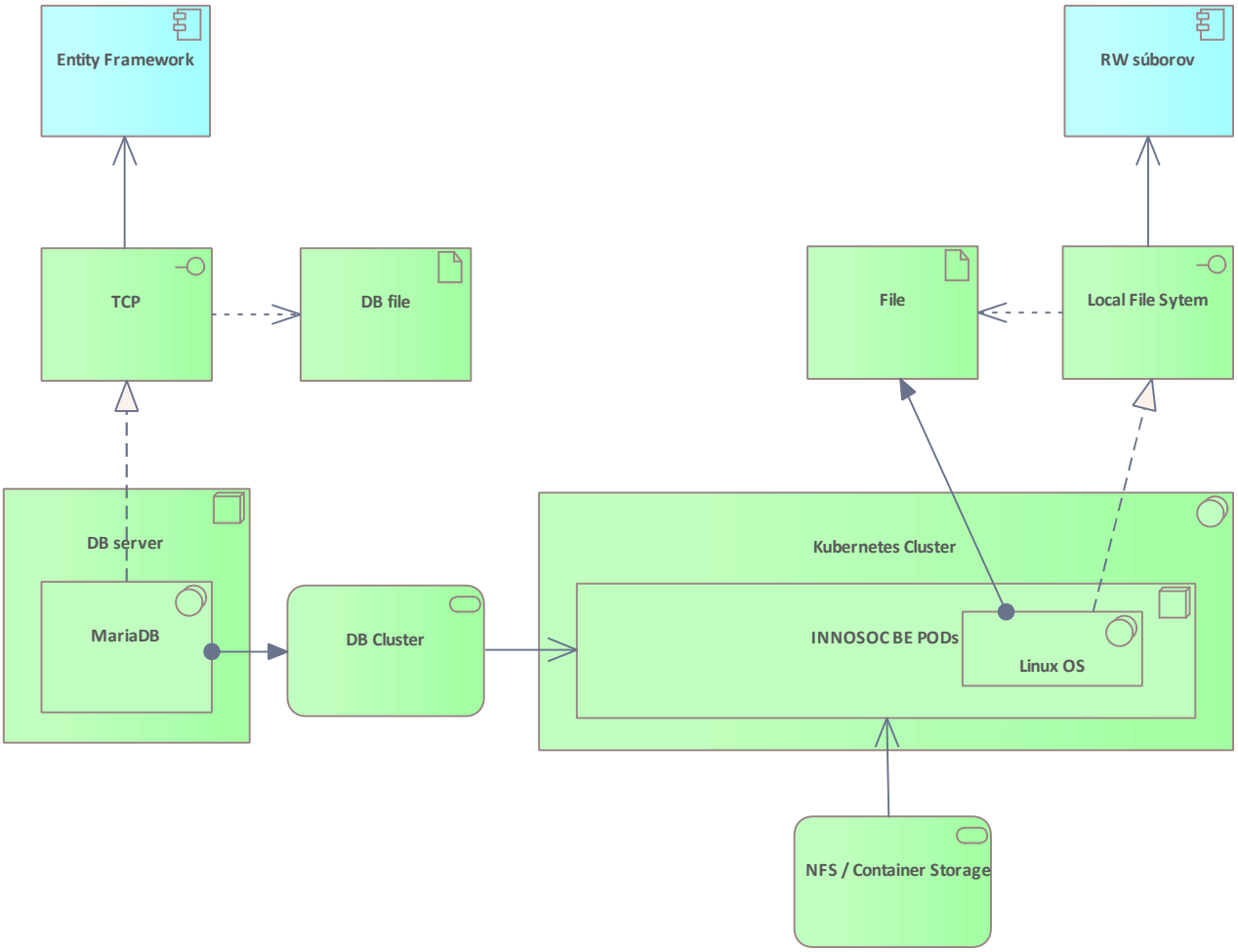


Figure 10: Technology Usage Viewpoint BE, DB and File System

The diagram represents and describes in more detail the use of the persistence layer for the needs of BIT NC and the BE module, by means of its Data Layer component, which handles working with PostgreSQL and the Local File System.

Description of the elements used:

DB Cluster

Represents a clustered relational database service. A clustered relational database service is a distributed database system that provides relational database services through multiple interconnected database nodes (clusters), thereby ensuring high availability, scalability, redundancy and fault tolerance.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms

and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

DB file

DB file – a file representing a MariaDB database file. A DB file (database file) in the context of the MariaDB system is a physical file in storage that represents the persistent data structure of the database. This file contains the data itself, metadata, indexes, transaction logs or system structures used by the database engine (most commonly InnoDB or MyISAM) to manage relational data.

DB server

Represents the MariaDB server. MariaDB Server is an open-source relational database management system (RDBMS) intended for managing, storing and manipulating structured data. It is compatible with MySQL (of which it is originally a fork), uses the SQL language and is optimized for performance, security, transactional integrity and scalability.

File

File – a file is the basic unit of data storage in a file system, representing structured or unstructured data such as attachments, images, documents, multimedia and other digital objects. A file contains metadata (e.g. name, type, size, timestamps, access rights) and binary or text content, which is interpreted according to its MIME type or extension.

Entity Framework

The Entity Framework (EF) application module represents a logical software layer in the application that uses Entity Framework technology for object-relational mapping (ORM) between the database and objects of the programming language (most commonly C# in the .NET ecosystem). This module ensures that the application can work with database tables as objects and collections of classes, with all CRUD operations (Create, Read, Update, Delete) automatically translated into SQL statements.

File RW

The file RW module represents a software component or logical layer that provides reading (Read) and writing (Write) of data to files within the file system. This module serves as an abstraction between the application logic and the physical storage, using the standard system interfaces of the operating system for file manipulation (e.g. POSIX API or abstract I/O libraries).

Local File Sytem

A local file system providing work with files, implemented by the operating system under which the BE module runs. This file system enables low-latency access to files, typically through standard POSIX operations, and is bound to a specific physical or virtual node. In containerized environments (e.g. Kubernetes), this type of storage is usually represented by the hostPath or emptyDir type, which binds it to a specific node and causes it to lose persistence when the POD is restarted or moved. A local file system is suitable for temporary,

high-performance operations (e.g. cache, scratch space), but is not intended for permanent or shared data storage, as it does not allow independent data recovery after an outage or scaling across nodes.

TCP

Represents client-server communication using the TCP protocol. Client-server communication over TCP (Transmission Control Protocol) is a method of exchanging data between two entities in a computer network, where the client initiates the connection and the server passively waits for a connection. The transfer is carried out by means of the TCP protocol – a transport protocol of layer 4 (transport layer) of the OSI model, which provides reliable, controlled and connection-oriented data transfer over the network.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

INNOSOC BE PODs

Represents the BE PODs. BE PODs refer to container instances of the system's backend services (BE – Backend) deployed in a distributed environment, typically in a Kubernetes cluster or other container orchestration. These PODs form the server (logical) layer of the architecture, which provides data processing, business logic, transactions, API services and interaction with the database and network entities (core network functions).

Linux OS

Linux is a multitasking, multi-user, Unix-compatible operating system (OS) built on a monolithic kernel and developed as open-source software. It is characterized by a modular architecture and high customizability, and is used in servers, desktops, mobile devices, embedded systems and supercomputers. Technically, Linux is a software layer between the hardware and user applications that provides management of processes, memory, file systems, devices and network communication.

MariaDB

Represents the MariaDB relational database. MariaDB is a relational database (Relational Database Management System – RDBMS). In terms of a formal definition, MariaDB is a database system that: works with the relational model (data are organized into tables, columns and rows), uses the SQL (Structured Query Language) language for data manipulation and querying, ensures data integrity by means of primary and foreign keys, and implements ACID properties through its storage engines.

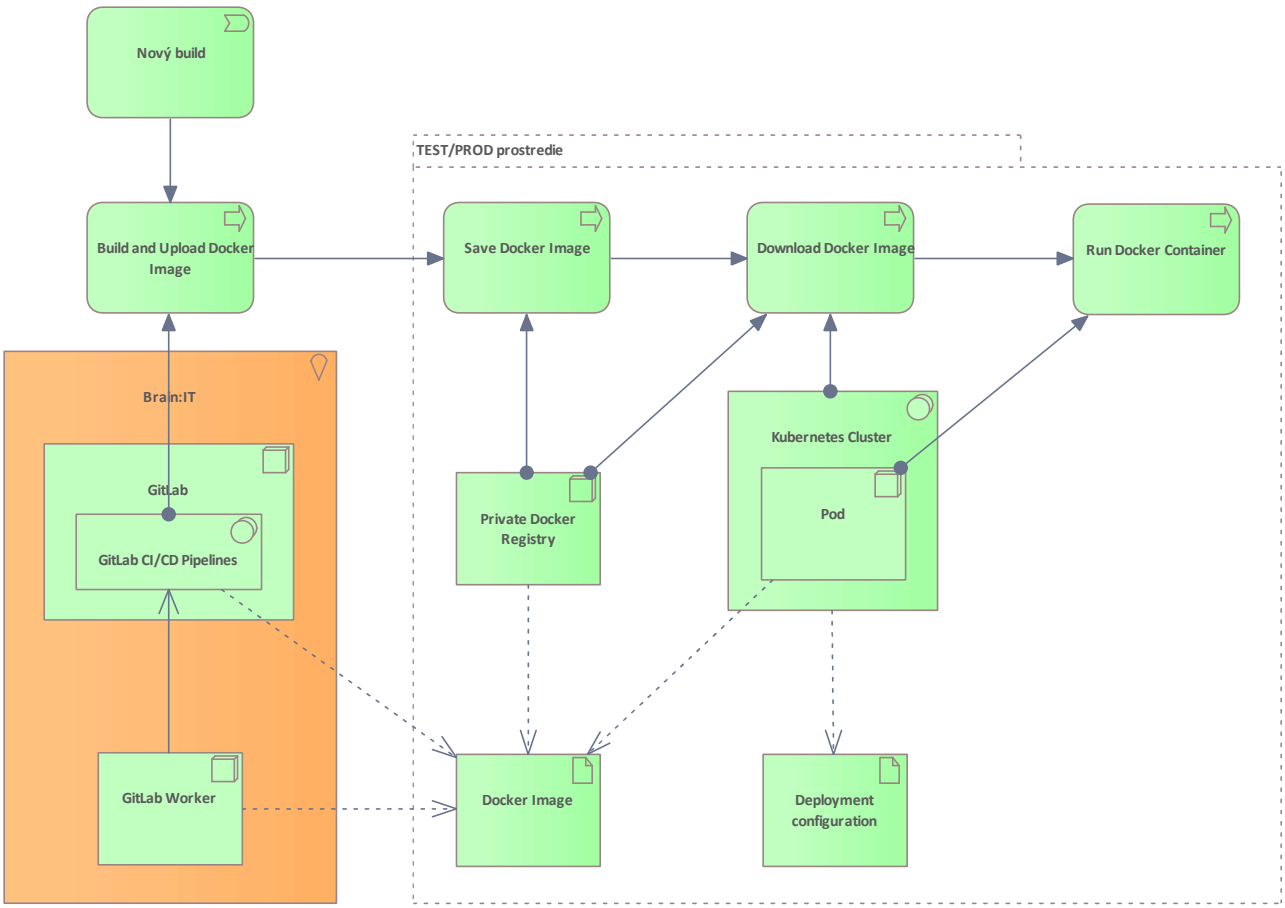


Figure 11: Technology Viewpoint Development CI/CD TEST/PROD

The diagram shows the process of building and deploying a new version of a package in the form of a Docker image to the TEST/PROD environment. The diagram is applicable to the case of in-house packages; for other existing ones, the public DockerHub source is used.

Description of the elements used:

Build and Upload Docker Image

The build process followed by uploading the Docker Image to the Docker Registry is a standard step that ensures automated assembly of the application container, its tagging and publication to a centralized repository, from which it can be deployed to development, testing or production environments.

Download Docker Image

Storing a Docker image in the Docker Registry begins with creating the image from a Dockerfile using the docker build command. The image is then tagged according to the name and version of the target registry. After authentication, the image is uploaded with the docker push command and becomes available for further deployment.

New build

Starting a new build process represents the initialization and configuration of an automated chain of tasks whose aim is to assemble, test and prepare a software artefact for further processing or deployment.

Run Docker Container

Running a Docker container begins with downloading or selecting an existing Docker image, from which an application instance is created. The container is started with the docker run command, where parameters such as ports, volume mounts, environment variables or network settings can be defined. Once started, the container runs as an isolated process, ready to communicate with services or the user.

Save Docker Image

The process of storing a Docker image in the Docker Registry includes creating the image using the docker build command and tagging it with the target registry address and version. Authentication to the registry then takes place, followed by the actual upload via the docker push command. As a result, the image is stored in a central repository, from which it can be deployed to any environment.

TEST/PROD environment

Represents the internal DEV and TEST environment

Brain:IT

Brain:IT site

Private Docker Registry

Represents a private Docker Registry. A Private Docker Registry is a private repository in which an organization securely stores and manages its own Docker images outside public registries such as Docker Hub. It enables full control over access, versioning, retention policies and security scanning of containers.

Deployment configuration

Representation of the Kubernetes Deployment artefact. A Kubernetes Deployment artefact is a configuration file that defines the desired state of an application deployment in a Kubernetes cluster – including the number of replicas, the Docker image, ports and update strategies. This artefact serves as input for kubectl or a CI/CD pipeline, which apply it to the cluster to ensure declarative and repeatable deployment.

Docker Image

Representation of the Docker Image artefact. A Docker Image as an artefact is the resulting application package assembled during the build process, which contains all the necessary layers, dependencies and configuration for running in a container environment. This artefact is stored in the Docker Registry, from which it can be reused for deployment in various environments.

GitLab

Represents a GitLab installation. A GitLab instance represents a standalone deployment of the GitLab platform, which can run as a cloud service (GitLab.com) or as a self-managed version on one's own infrastructure. It contains all the components needed for repository management, CI/CD pipelines, issue tracking and DevOps workflows in a single integrated environment. Within an organization, it enables full control over projects, users, security and access rights.

GitLab Worker

Represents a GitLab Worker instance. A GitLab Worker instance (also called GitLab Runner) is a standalone service that executes tasks defined in a CI/CD pipeline, such as building, testing or deploying applications. It runs in a dedicated environment (Kubernetes) and communicates with the central GitLab server, from which it receives jobs for processing.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

GitLab CI/CD Pipelines

GitLab CI/CD Pipelines are a series of tasks providing compilation, tests, configuration, deployment and other activities within GitLab.

Pod

Represents a Pod instance in which a new Docker Image is run in the form of a Docker Container

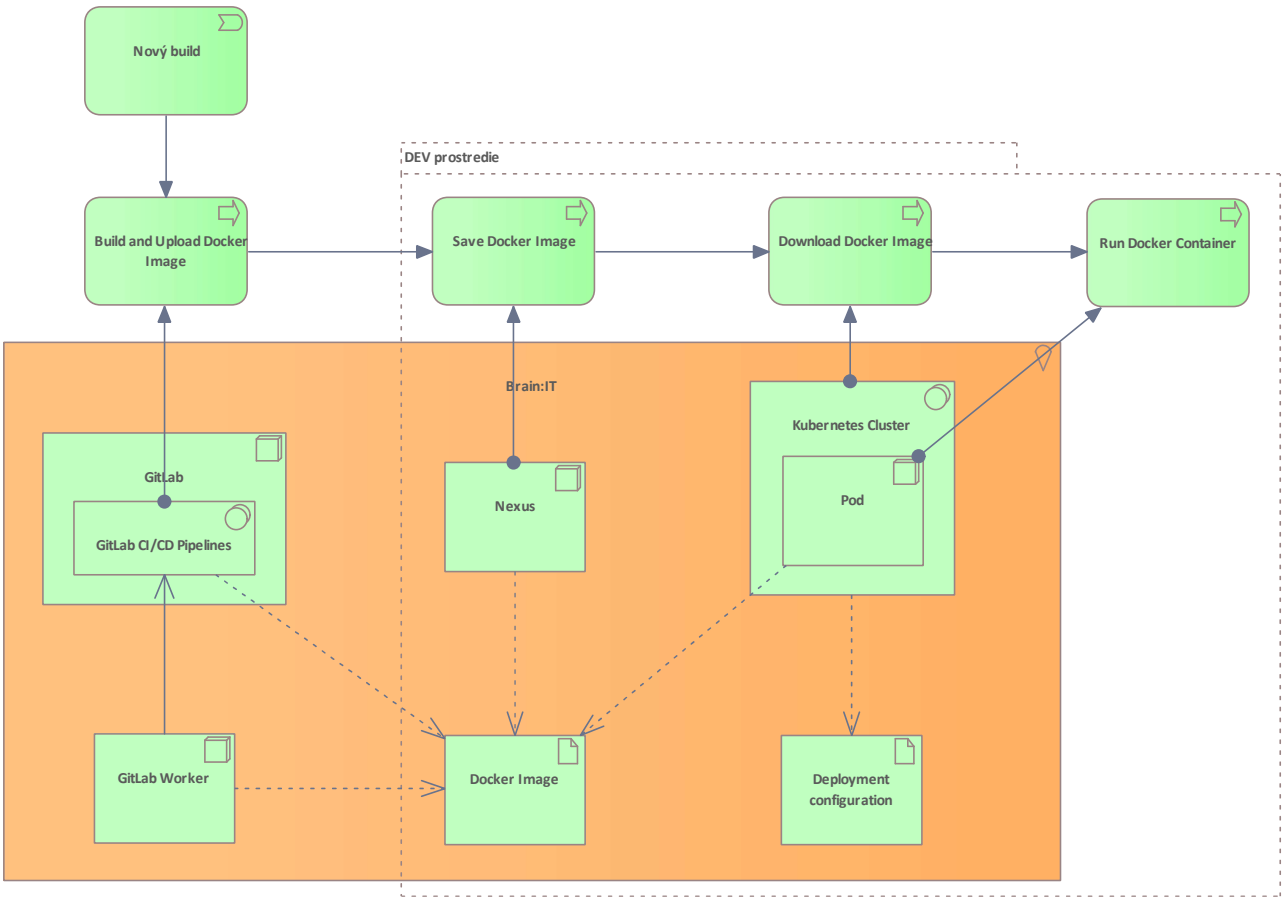


Figure 12: Technology Viewpoint Developmnent CI/CD DEV

The diagram shows the process of building and deploying a new version of a package in the form of a Docker image to the DEV environment. The diagram is applicable to the case of in-house packages; for other existing ones, the public DockerHub source is used.

Description of the elements used:

Build and Upload Docker Image

The build process followed by uploading the Docker Image to the Docker Registry is a standard step that ensures automated assembly of the application container, its tagging and publication to a centralized repository, from which it can be deployed to development, testing or production environments.

Download Docker Image

Storing a Docker image in the Docker Registry begins with creating the image from a Dockerfile using the docker build command. The image is then tagged according to the name and version of the target registry. After authentication, the image is uploaded with the docker push command and becomes available for further deployment.

New build

Starting a new build process represents the initialization and configuration of an automated chain of tasks whose aim is to assemble, test and prepare a software artefact for further processing or deployment.

Run Docker Container

Running a Docker container begins with downloading or selecting an existing Docker image, from which an application instance is created. The container is started with the docker run command, where parameters such as ports, volume mounts, environment variables or network settings can be defined. Once started, the container runs as an isolated process, ready to communicate with services or the user.

Save Docker Image

The process of storing a Docker image in the Docker Registry includes creating the image using the docker build command and tagging it with the target registry address and version. Authentication to the registry then takes place, followed by the actual upload via the docker push command. As a result, the image is stored in a central repository, from which it can be deployed to any environment.

DEV environment

Represents the internal DEV and TEST environment

Brain:IT

Brain:IT site

Deployment configuration

Representation of the Kubernetes Deployment artefact. A Kubernetes Deployment artefact is a configuration file that defines the desired state of an application deployment in a Kubernetes cluster – including the number of replicas, the Docker image, ports and update strategies. This artefact serves as input for kubectl or a CI/CD pipeline, which apply it to the cluster to ensure declarative and repeatable deployment.

Docker Image

Representation of the Docker Image artefact. A Docker Image as an artefact is the resulting application package assembled during the build process, which contains all the necessary layers, dependencies and configuration for running in a container environment. This artefact is stored in the Docker Registry, from which it can be reused for deployment in various environments.

GitLab

Represents a GitLab installation. A GitLab instance represents a standalone deployment of the GitLab platform, which can run as a cloud service (GitLab.com) or as a self-

managed version on one's own infrastructure. It contains all the components needed for repository management, CI/CD pipelines, issue tracking and DevOps workflows in a single integrated environment. Within an organization, it enables full control over projects, users, security and access rights.

GitLab Worker

Represents a GitLab Worker instance. A GitLab Worker instance (also called GitLab Runner) is a standalone service that executes tasks defined in a CI/CD pipeline, such as building, testing or deploying applications. It runs in a dedicated environment (Kubernetes) and communicates with the central GitLab server, from which it receives jobs for processing.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Nexus

Represents a private Docker Registry. A Private Docker Registry is private storage for Docker images that enables organizations to securely store and manage container images outside public services such as Docker Hub. It provides control over access, versions and distribution of images, and is often integrated with a CI/CD pipeline for automated builds and deployments.

GitLab CI/CD Pipelines

GitLab CI/CD Pipelines are a series of tasks providing compilation, tests, configuration, deployment and other activities within GitLab.

Pod

Represents a Pod instance in which a new Docker Image is run in the form of a Docker Container

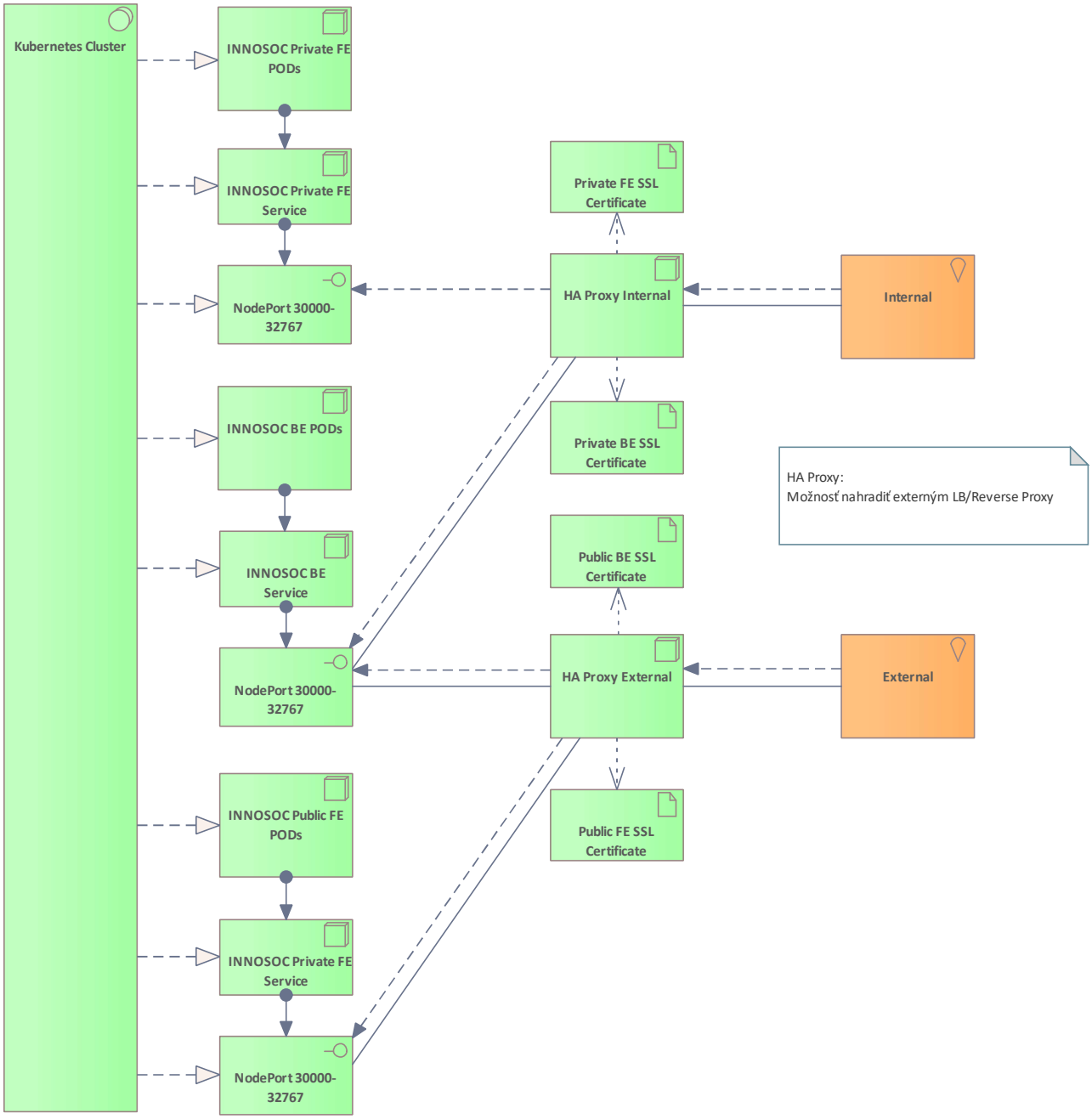


Figure 13: Technology Viewpoint Interconnect

The diagram shows the deployment and interconnection of exposed Kubernetes services and the LB/Reverse Proxy for two zones.

Description of the elements used:

External

Represents the external Internet environment

HA Proxy External

Represents an external reverse proxy built on HA Proxy. An external reverse proxy built on HAProxy is an intermediary layer located at the network edge that serves to receive requests from the internet, which it then redirects to internal backend services. In this role, HAProxy provides TLS termination, load balancing, protection through ACL rules, and also the possibility of filtering or passing traffic based on defined security policies. This architecture increases the security, scalability and availability of publicly accessible services, while separating the external layer from the internal infrastructure.

HA Proxy Internal

Represents an internal reverse proxy built on HA Proxy. An internal reverse proxy built on HAProxy is an intermediary service that receives incoming requests within the internal network and redirects them to the relevant backend services according to configured rules. In this case, HAProxy provides load balancing, TLS termination, health checking and the possibility of centrally controlling the data flow between microservices or internal applications.

INNOSOC BE Service

Represents the INNOSOC BE Service. It runs as part of a preconfigured POD, often in the form of a container or virtual machine, and is updated through a controlled CI/CD process.

INNOSOC Private FE Service

Represents the INNOSOC FE Private Service. It runs as part of a preconfigured POD, often in the form of a container or virtual machine, and is updated through a controlled CI/CD process.

INNOSOC Private FE Service

Represents the INNOSOC FE Public Service. It runs as part of a preconfigured POD, often in the form of a container or virtual machine, and is updated through a controlled CI/CD process.

Internal

Represents the internal BRAIN:IT environment

Private BE SSL Certificate

Represents a valid Private SSL certificate for HTTPS purposes. A valid BE SSL certificate for HTTPS purposes is a security certificate installed on the backend (BE) layer of the system that enables encrypted and authenticated communication between services or between the frontend and the backend. The certificate must be issued by a trusted certification authority (CA) or by an internal PKI infrastructure and must ensure the integrity and confidentiality of data transmitted over HTTPS.

Private FE SSL Certificate

Represents a valid SSL certificate for HTTPS purposes. A valid FE SSL certificate for HTTPS purposes is a trusted digital certificate installed on the frontend (FE) layer of the application that ensures encrypted communication between the user's browser and the frontend server. This certificate must be issued by an accredited certification authority (CA), must be valid for the specific domain and must not be expired or invalid.

Public BE SSL Certificate

Represents a valid Public SSL certificate for HTTPS purposes. A valid BE Public SSL certificate for HTTPS purposes is a certificate installed on a backend service that is accessible from the public network, and it serves to ensure encrypted and verified communication over HTTPS. It must be issued by a trusted public certification authority (CA), be valid for the specific public domain and meet security standards.

Public FE SSL Certificate

Represents a valid Public SSL certificate for HTTPS purposes. A valid FE Public SSL certificate for HTTPS purposes is a trusted certificate installed on the frontend layer of an application that is publicly accessible via the internet, and it ensures encrypted communication between the client browser and the frontend server. It must be issued by a public certification authority (CA), be valid for the given domain and must not be expired or revoked in order to be accepted by modern browsers.

NodePort 30000-32767

Represents NodePort-type routing from the range 30000-32767

NodePort 30000-32767

Represents NodePort-type routing from the range 30000-32767

NodePort 30000-32767

Represents NodePort-type routing from the range 30000-32767

Note

HA Proxy: Option to replace with an external LB/Reverse Proxy

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes

cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

INNOSOC BE PODs

Represents the BE PODs. BE PODs refer to container instances of the system's backend services (BE – Backend) deployed in a distributed environment, typically in a Kubernetes cluster or other container orchestration. These PODs form the server (logical) layer of the architecture, which provides data processing, business logic, transactions, API services and interaction with the database and network entities (core network functions).

INNOSOC Private FE PODs

FE Private PODs represent isolated and modular deployment units of the system's frontend components in the customer's private environment. This is an architectural element based on the principles of containerization and microservice architecture, where each POD contains independently deployable elements of the presentation layer (FE – frontend), such as web applications, Single Page Applications (SPA) or UI modules.

INNOSOC Public FE PODs

FE Public PODs represent computing deployment units of the system's frontend components in a publicly accessible cloud or hybrid environment managed by the service provider or by the organization's centralized infrastructure. Within this architecture, the frontend components – typically Single Page Applications (SPA) or web interfaces – are operated as part of a multi-tenant system accessible via a public network (the internet), with communication and access governed by strict security policies. These PODs function as isolated deployments within a centralized cloud (e.g. Microsoft Azure, AWS, GCP) or in a managed Kubernetes cluster. Their aim is to provide a scalable, highly available and easily updatable interface for end users or customers who access the services without the need for a local installation or private infrastructure.

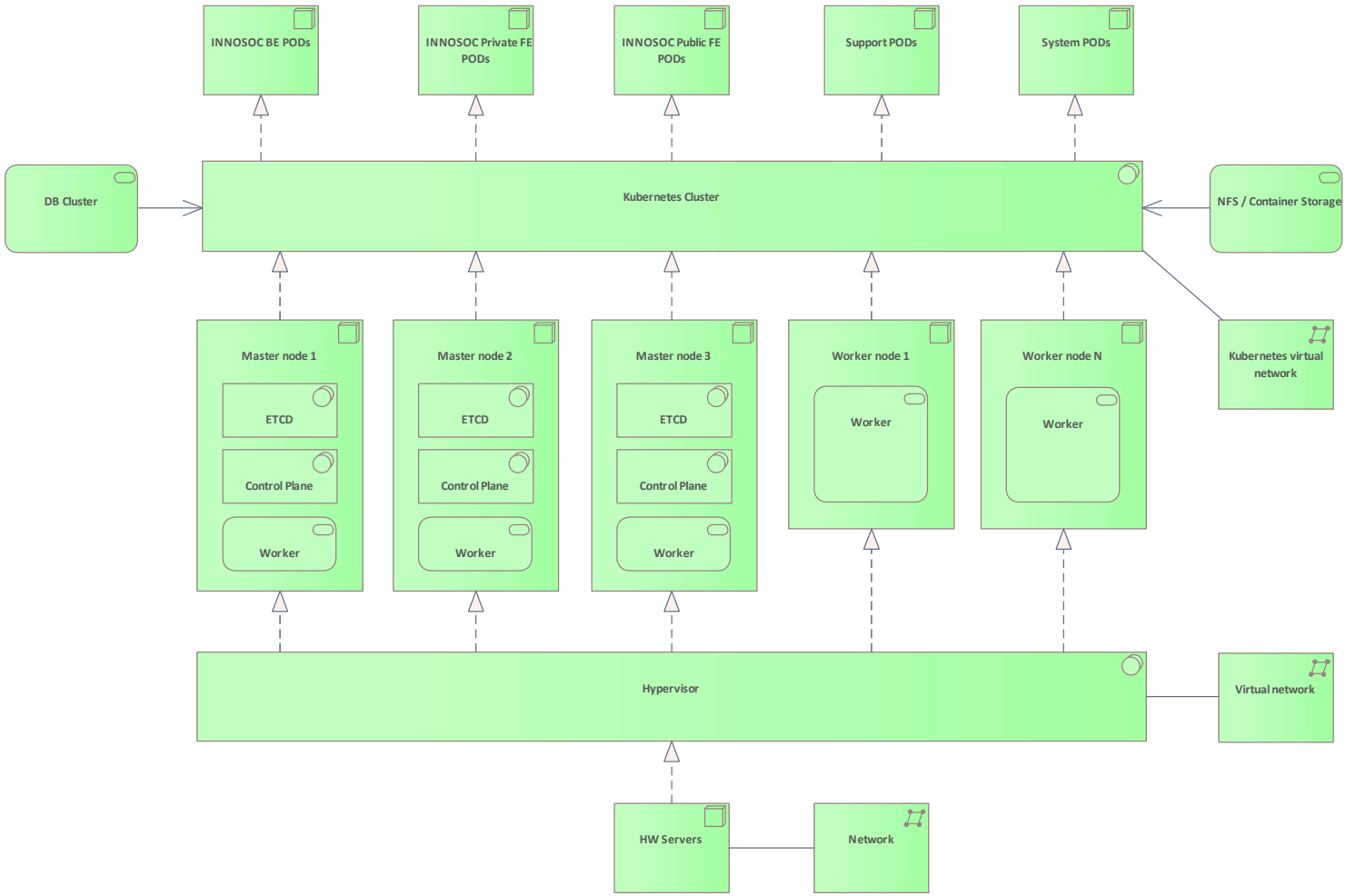


Figure 14: Technology Viewpoint Kubernetes Cluster

Represents the role and service of the Worker node.

Description of the elements used:

DB Cluster

Represents a clustered relational database service. A clustered relational database service is a distributed database system that provides relational database services through multiple interconnected database nodes (clusters), thereby ensuring high availability, scalability, redundancy and fault tolerance.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

HW Servers

Represents the HW server infrastructure. The HW server infrastructure represents the physical computing and network equipment that forms the basis for operating IT systems, including servers, storage, switches, routers and backup power supplies. This infrastructure is deployed either on-premise or as part of a cloud platform, and provides computing power, availability and connectivity for both virtualized and non-virtualized environments.

Hypervisor

Represents a Type 1 or Type 2 Hypervisor. A Type 1 hypervisor, also known as a "bare-metal" hypervisor, runs directly on the physical hardware without an operating system, thereby providing high performance and efficiency of virtualization; examples include VMware ESXi, Microsoft Hyper-V (core) or Xen. This type of hypervisor is intended mainly for data centres and enterprise environments where emphasis is placed on performance, stability and security.

A Type 2 hypervisor, also called a "hosted" hypervisor, runs on top of an existing operating system as an application, and thus shares system resources with the host; examples include VMware Workstation or Oracle VirtualBox. Although it is simpler to install and suitable for development or testing, its performance and isolation are lower compared with Type 1.

Kubernetes virtual network

Represents the virtual network/subnet of the Kubernetes Cluster. A virtual network or subnet in a Kubernetes cluster is a logical network architecture that enables communication between pods, services and other components in the cluster without the need for NAT. Each pod receives its own IP address within this network, which ensures predictable and fully routable connectivity in accordance with the principle that "every pod can communicate with every other pod".

Master node 1

Represents the Master node of the Kubernetes Cluster. A Master node in a Kubernetes cluster is a controller node that hosts the Control Plane components responsible for global decision-making, workload scheduling and maintaining the cluster state. It contains key services such as kube-apiserver, the ETCD database, the scheduler and controller managers, which together provide orchestration and management of all Worker nodes and pods.

Master node 2

Represents the Master node of the Kubernetes Cluster. A Master node in a Kubernetes cluster is a controller node that hosts the Control Plane components responsible for global decision-making, workload scheduling and maintaining the cluster state. It contains key services such as kube-apiserver, the ETCD database, the scheduler and controller managers, which together provide orchestration and management of all Worker nodes and pods.

Master node 3

Represents the Master node of the Kubernetes Cluster. A Master node in a Kubernetes cluster is a controller node that hosts the Control Plane components responsible for global decision-making, workload scheduling and maintaining the cluster state. It contains key services such as kube-apiserver, the ETCD database, the scheduler and controller managers, which together provide orchestration and management of all Worker nodes and pods.

Network

Represents the network/subnet of the HW infrastructure. The network or subnet of the HW infrastructure is the physical and logical layer interconnecting servers, storage and network devices, which provides data transfer, communication and access to services within the data centre.

Support PODs

Represents the support service PODs. Support service PODs are container or virtual deployment units that provide non-application functions necessary for running and managing the main system, such as monitoring, logging, authentication or configuration. These services typically function as infrastructure components and are deployed in parallel with the application PODs within the same cluster or environment.

System PODs

Represents the system service PODs. System service PODs are deployed components in a containerized environment that provide the basic functionality and operations necessary for the correct operation of the entire platform or cluster, such as networking, DNS, storage or orchestration. These services usually form part of the core infrastructure of Kubernetes or another orchestrator and function independently of the application and support PODs, while ensuring the stability and operability of the system.

Virtual network

Represents the virtual network/subnet of the Hypervisor. The virtual network or subnet of the Hypervisor is a logical network infrastructure that the hypervisor creates for the purpose of interconnecting virtual machines (VMs) with one another, with the host system or with the external network. It enables the definition of virtual switches and NAT, bridged or internal networking modes, thereby providing flexibility in the isolation, communication and routing of data between VMs and the physical network environment.

Worker node 1

Represents a Worker node of the Kubernetes Cluster. A Worker node in a Kubernetes cluster is a compute node on which application Pods, i.e. the actual workloads of the system, are deployed and run. It contains basic components such as kubelet, kube-proxy and a container runtime (e.g. containerd), which ensure communication with the Control Plane and correct execution of the assigned tasks.

Worker node N

Represents a Worker node of the Kubernetes Cluster. A Worker node in a Kubernetes cluster is a compute node on which application Pods, i.e. the actual workloads of the system, are deployed and run. It contains basic components such as kubelet, kube-proxy and a container runtime (e.g. containerd), which ensure communication with the Control Plane and correct execution of the assigned tasks.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

INNOSOC BE PODs

Represents the BE PODs. BE PODs refer to container instances of the system's backend services (BE – Backend) deployed in a distributed environment, typically in a Kubernetes cluster or other container orchestration. These PODs form the server (logical) layer of the architecture, which provides data processing, business logic, transactions, API services and interaction with the database and network entities (core network functions).

INNOSOC Private FE PODs

FE Private PODs represent isolated and modular deployment units of the system's frontend components in the customer's private environment. This is an architectural element based on the principles of containerization and microservice architecture, where each POD contains independently deployable elements of the presentation layer (FE – frontend), such as web applications, Single Page Applications (SPA) or UI modules.

INNOSOC Public FE PODs

FE Public PODs represent computing deployment units of the system's frontend components in a publicly accessible cloud or hybrid environment managed by the service provider or by the organization's centralized infrastructure. Within this architecture, the frontend components – typically Single Page Applications (SPA) or web interfaces – are operated as part of a multi-tenant system accessible via a public network (the internet), with communication and access governed by strict security policies. These PODs function as isolated deployments within a centralized cloud (e.g. Microsoft Azure, AWS, GCP) or in a managed Kubernetes cluster. Their aim is to provide a scalable, highly available and easily updatable interface for end users or customers who access the services without the need for a local installation or private infrastructure.

Worker

Represents the role and service of the Worker node. The role and service of a Worker node in Kubernetes consists in hosting and running application pods, while ensuring the execution of workloads based on instructions from the Control Plane.

Control Plane

Represents the Control Plane for managing the Kubernetes Cluster. The Control Plane for managing a Kubernetes cluster is the central control layer that provides global orchestration, scheduling and management of all components in the cluster. It contains key services such as kube-apiserver, the ETCD database, kube-scheduler, kube-controller-manager and possibly cloud-controller-manager. The Control Plane decides where and how applications will run, monitors their state and guarantees that the desired state (declared by the user) corresponds to the current state of the cluster.

ETCD

Represents the ETCD database. The ETCD database is a distributed, highly available key-value database designed for storing configuration data and metadata in real time. It is a core part of Kubernetes, where it serves as the central store of cluster state, including information about pods, configurations, secrets and other objects. Thanks to the Raft consensus mechanism, it ensures reliability, consistency and fault tolerance in distributed systems.

Worker

Represents the role and service of the Worker node. The role and service of a Worker node in Kubernetes consists in hosting and running application pods, while ensuring the execution of workloads based on instructions from the Control Plane.

Control Plane

Represents the Control Plane for managing the Kubernetes Cluster. The Control Plane for managing a Kubernetes cluster is the central control layer that provides global orchestration, scheduling and management of all components in the cluster. It contains key services such as kube-apiserver, the ETCD database, kube-scheduler, kube-controller-

manager and possibly cloud-controller-manager. The Control Plane decides where and how applications will run, monitors their state and guarantees that the desired state (declared by the user) corresponds to the current state of the cluster.

ETCD

Represents the ETCD database. The ETCD database is a distributed, highly available key-value database designed for storing configuration data and metadata in real time. It is a core part of Kubernetes, where it serves as the central store of cluster state, including information about pods, configurations, secrets and other objects. Thanks to the Raft consensus mechanism, it ensures reliability, consistency and fault tolerance in distributed systems.

Worker

Represents the role and service of the Worker node. The role and service of a Worker node in Kubernetes consists in hosting and running application pods, while ensuring the execution of workloads based on instructions from the Control Plane.

Control Plane

Represents the Control Plane for managing the Kubernetes Cluster. The Control Plane for managing a Kubernetes cluster is the central control layer that provides global orchestration, scheduling and management of all components in the cluster. It contains key services such as kube-apiserver, the ETCD database, kube-scheduler, kube-controller-manager and possibly cloud-controller-manager. The Control Plane decides where and how applications will run, monitors their state and guarantees that the desired state (declared by the user) corresponds to the current state of the cluster.

ETCD

Represents the ETCD database. The ETCD database is a distributed, highly available key-value database designed for storing configuration data and metadata in real time. It is a core part of Kubernetes, where it serves as the central store of cluster state, including information about pods, configurations, secrets and other objects. Thanks to the Raft consensus mechanism, it ensures reliability, consistency and fault tolerance in distributed systems.

Worker

Represents the role and service of the Worker node. The role and service of a Worker node in Kubernetes consists in hosting and running application pods, while ensuring the execution of workloads based on instructions from the Control Plane.

Worker

Represents the role and service of the Worker node. The role and service of a Worker node in Kubernetes consists in hosting and running application pods, while ensuring the execution of workloads based on instructions from the Control Plane.

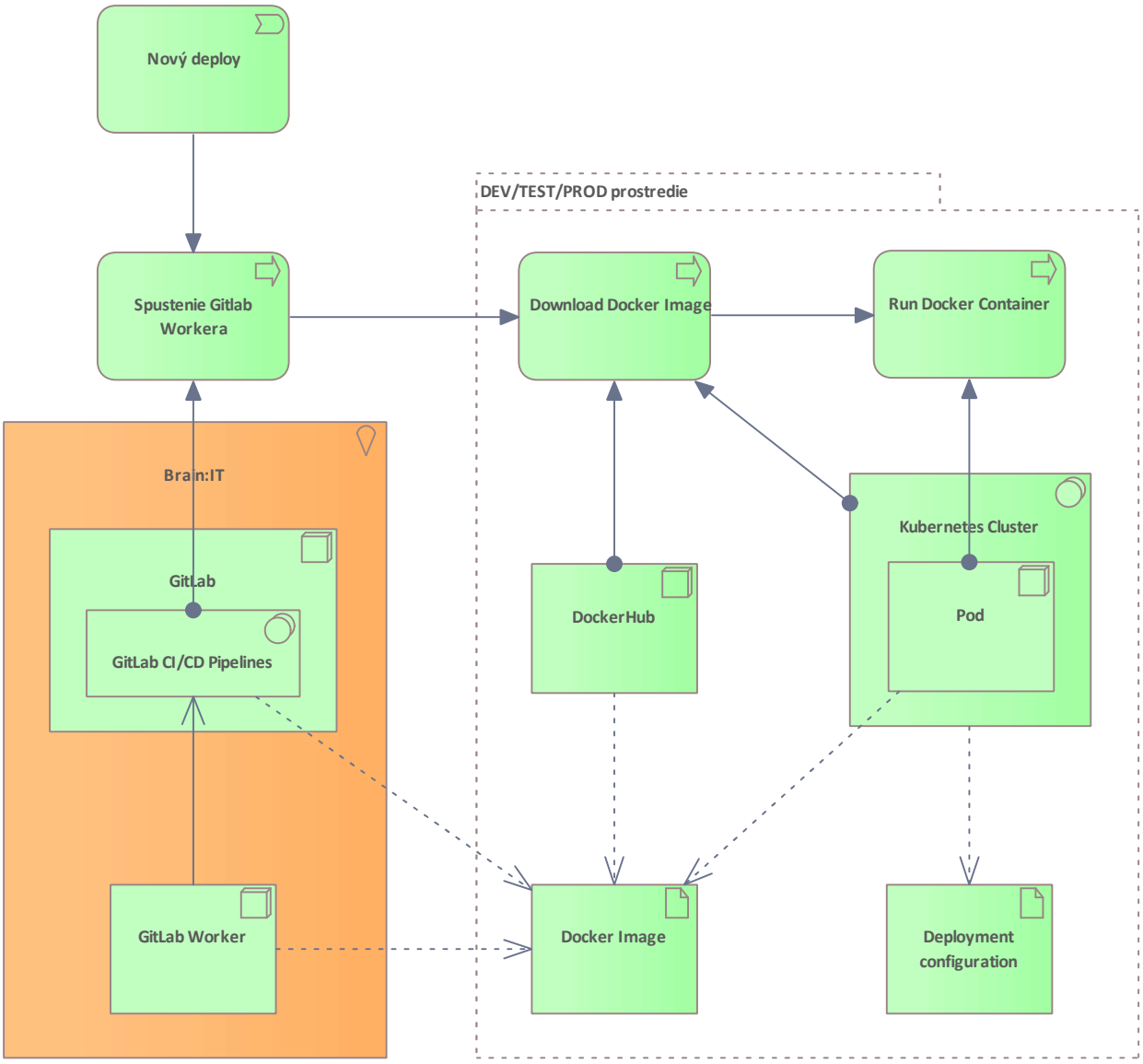


Figure 15: Technology Viewpoint standard CI/CD DEV/TEST/PROD

Description of the elements used:

Download Docker Image

Storing a Docker image in the Docker Registry begins with creating the image from a Dockerfile using the docker build command. The image is then tagged according to the name and version of the target registry. After authentication, the image is uploaded with the docker push command and becomes available for further deployment.

New deploy

Starting a new deploy process is an action in which a new version of an application or service is automatically deployed to the target environment on the basis of a defined CI/CD pipeline.

Run Docker Container

Running a Docker container begins with downloading or selecting an existing Docker image, from which an application instance is created. The container is started with the docker run command, where parameters such as ports, volume mounts, environment variables or network settings can be defined. Once started, the container runs as an isolated process, ready to communicate with services or the user.

Starting the Gitlab Worker

The process of starting a GitLab Worker to deploy a new version of a package begins with receiving a job from the CI/CD pipeline defined in .gitlab-ci.yml, which contains the deployment steps. The Worker then executes instructions such as downloading the artefact, connecting to the target environment and deploying the new version of the application, with the entire run being logged and traceable.

DEV/TEST/PROD environment

Represents the internal DEV and TEST environment

Brain:IT

Brain:IT site

DockerHub

Represents the public Docker Registry DockerHub. DockerHub is a public Docker Registry provided by Docker Inc. that enables developers to share, store and download container images via a cloud platform. It supports both public and private repositories, image versioning, automated builds and integration with CI/CD tools.

Deployment configuration

Representation of the Kubernetes Deployment artefact. A Kubernetes Deployment artefact is a configuration file that defines the desired state of an application deployment in a Kubernetes cluster – including the number of replicas, the Docker image, ports and update strategies. This artefact serves as input for kubectl or a CI/CD pipeline, which apply it to the cluster to ensure declarative and repeatable deployment.

Docker Image

Representation of the Docker Image artefact. A Docker Image as an artefact is the resulting application package assembled during the build process, which contains all the necessary layers, dependencies and configuration for running in a container environment. This artefact is stored in the Docker Registry, from which it can be reused for deployment in various environments.

GitLab

Represents a GitLab installation. A GitLab instance represents a standalone deployment of the GitLab platform, which can run as a cloud service (GitLab.com) or as a self-managed version on one's own infrastructure. It contains all the components needed for repository management, CI/CD pipelines, issue tracking and DevOps workflows in a single integrated environment. Within an organization, it enables full control over projects, users, security and access rights.

GitLab Worker

Represents a GitLab Worker instance. A GitLab Worker instance (also called GitLab Runner) is a standalone service that executes tasks defined in a CI/CD pipeline, such as building, testing or deploying applications. It runs in a dedicated environment (Kubernetes) and communicates with the central GitLab server, from which it receives jobs for processing.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

GitLab CI/CD Pipelines

GitLab CI/CD Pipelines are a series of tasks providing compilation, tests, configuration, deployment and other activities within GitLab.

Pod

Represents a Pod instance in which a new Docker Image is run in the form of a Docker Container

1.1.6. Technology Viewpoint OpenStack

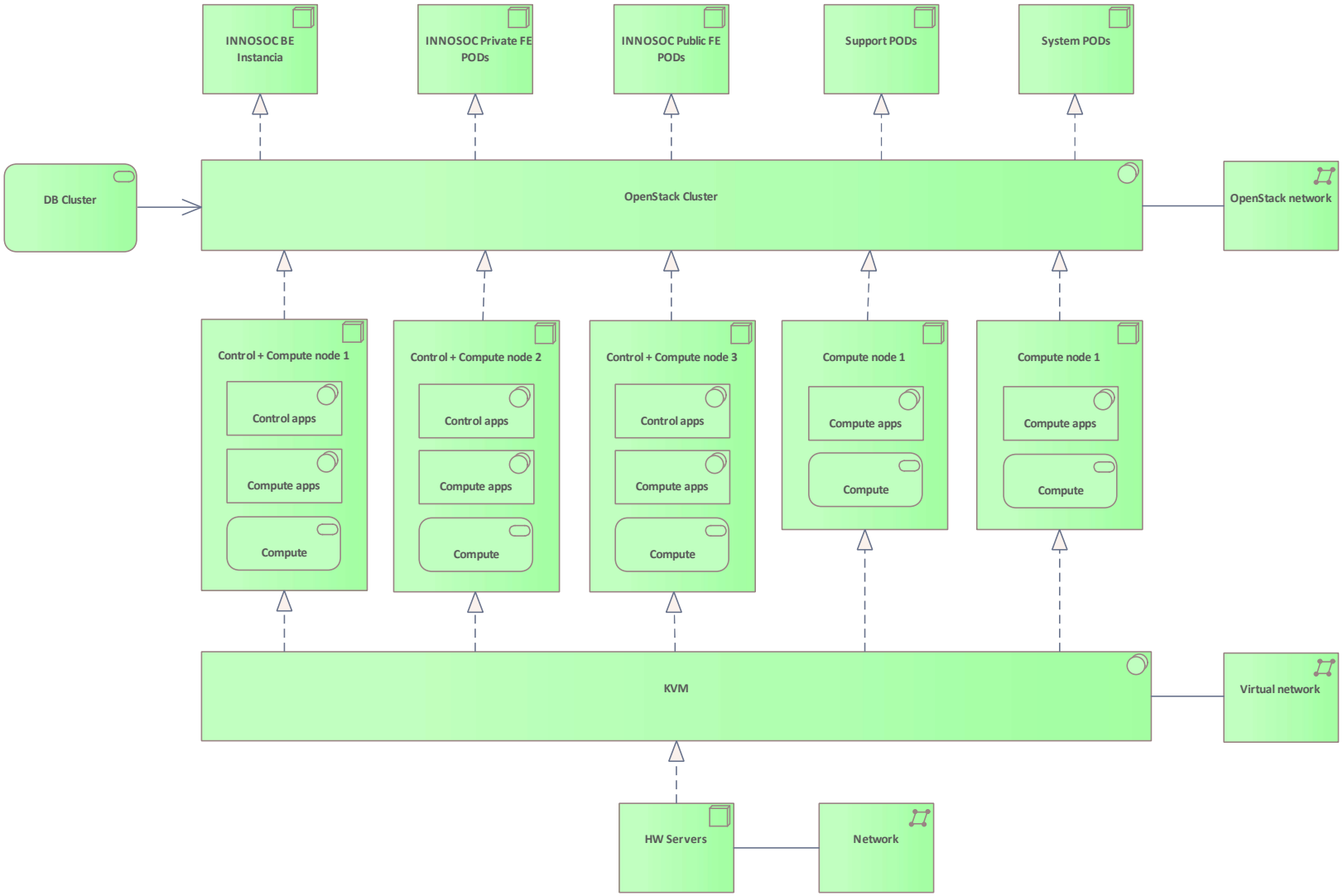


Figure 16: Technology Viewpoint OpenStack Cluster

Description of the elements used:

DB Cluster

Represents a clustered relational database service. A clustered relational database service represents a multi-node database solution in which data are partitioned or replicated across several database nodes in order to ensure high availability, scalability and fault tolerance. Each node in the cluster can serve as an active or passive member, depending on the mode used. This type of database architecture is used in enterprise systems where data consistency, performance under load and continuous availability are critical.

Compute node 1

Represents a Compute node of the OpenStack Cluster. A Compute node in an OpenStack cluster is a compute node responsible for running and managing virtual machines (VMs) using the Nova Compute service. It receives instructions from the Control node and provides physical or virtual resources such as CPU, RAM and disk for the individual instances, thereby forming the basis of the cluster's computing capacity.

Compute node 1

Represents a Compute node of the OpenStack Cluster. A Compute node in an OpenStack cluster is a compute node responsible for running and managing virtual machines (VMs) using the Nova Compute service. It receives instructions from the Control node and provides physical or virtual resources such as CPU, RAM and disk for the individual instances, thereby forming the basis of the cluster's computing capacity.

Control + Compute node 1

Control and Compute nodes in an OpenStack cluster are the basic node types that together provide the control and operation of the cloud environment. The Control node contains central services such as API servers, schedulers (Scheduler), the database, a message broker and authentication, thereby controlling orchestration, resource management and communication within the cluster. Compute node 1 provides computing capacity for running virtual machines (VMs) and ensures their local management using the Nova Compute component, according to instructions from the Control node.

Control + Compute node 2

Control and Compute nodes in an OpenStack cluster are the basic node types that together provide the control and operation of the cloud environment. The Control node contains central services such as API servers, schedulers (Scheduler), the database, a message broker and authentication, thereby controlling orchestration, resource management and communication within the cluster. Compute node 2 provides computing capacity for running virtual machines (VMs) and ensures their local management using the Nova Compute component, according to instructions from the Control node.

Control + Compute node 3

Control and Compute nodes in an OpenStack cluster are the basic node types that together provide the control and operation of the cloud environment. The Control node contains central services such as API servers, schedulers (Scheduler), the database, a message broker

and authentication, thereby controlling orchestration, resource management and communication within the cluster. Compute node 3 provides computing capacity for running virtual machines (VMs) and ensures their local management using the Nova Compute component, according to instructions from the Control node.

HW Servers

The HW server infrastructure is the physical foundation of the IT environment, comprising compute servers, storage, network elements, power supplies and supporting equipment, which together provide the necessary performance, availability and connectivity for the operation of information systems. This infrastructure forms the platform for virtualization, cloud services, databases and application workloads in both enterprise and data centres.

KVM

KVM (Kernel-based Virtual Machine) is a Type 1 hypervisor that is integrated directly into the Linux kernel and enables multiple virtual machines to run with full isolation on a physical server. It uses hardware virtualization and provides high performance, stability and efficiency, which makes it suitable for production and cloud environments such as OpenStack.

Network

Represents the network/subnet of the HW infrastructure. The network or subnet of the HW infrastructure represents the physical and logical interconnection between servers, storage, network devices and other components within the data centre. It serves to ensure data transfer, traffic segregation and control of communication by means of routers, switches, VLANs and security rules.

OpenStack Cluster

Represents an OpenStack cluster, which is a distributed cloud environment consisting of multiple physical or virtual nodes that jointly operate the individual OpenStack services, such as compute (Nova), networking (Neutron), storage (Cinder, Swift) and control components. This cluster enables organizations to build a private or hybrid cloud with support for automated provisioning, resource management and a scalable multi-tenant architecture.

OpenStack network

The network and subnets of the OpenStack cluster serve for the logical and physical separation of different types of communication within the cloud infrastructure, with each performing a specific role. The Corporate network provides the interconnection of the OpenStack environment with the enterprise infrastructure, the OpenStack Tenant Data Network provides network connectivity for the virtual instances of the individual tenants, and the OpenStack Internal Admin / API Subnet is reserved for internal communication between cluster components, including API calls, database replication and service management.

Support PODs

Represents the support service PODs. Support service PODs are container units that provide non-functional, infrastructure or operational functions necessary for running the

application system, such as monitoring, logging, authentication or configuration. These services run independently of the application PODs and support the stability, security and management of the entire environment.

System PODs

Represents the system service PODs. System service PODs are basic container units that provide key functionalities of the Kubernetes cluster and its infrastructure, such as network connectivity, DNS, workload scheduling or access control. They typically include components such as CoreDNS, kube-proxy, metrics-server or other system operators necessary for the correct functioning of the environment. These PODs are critical for the stability and operability of the platform and run at the level of system namespaces outside the application layer.

Virtual network

Represents the virtual network/subnet of the Hypervisor. The virtual network or subnet of the hypervisor is a logical network layer that enables communication between virtual machines, with the host or with an external network. Within this network, the hypervisor creates and manages virtual switches, bridges or NAT interfaces, thereby ensuring flexible routing, isolation and control of network traffic between VMs and the physical infrastructure.

INNOSOC BE Instance

Represents an OpenStack instance, a VM or a container. An OpenStack instance in the form of a VM or a container represents a virtual computing resource (compute instance) deployed within the OpenStack cloud platform by means of the Nova component. It may be a classic virtual machine (VM) or a lighter container instance, depending on the hypervisor used and the integration with containerization technologies such as Kubernetes or Docker.

INNOSOC Private FE PODs

Represents the FE Private PODs. FE Private PODs are isolated container deployment units of the system's frontend layer that are operated in the customer's private environment, for example on-premise or in a private cloud. These PODs ensure secure and separate provision of the user interface, while being integrated with the customer's local infrastructure, authentication and network policies.

INNOSOC Public FE PODs

Represents the FE Public PODs. FE Public PODs are container deployment units of the system's frontend layer that are operated in a publicly available cloud environment and accessible via the internet. They serve to provide the user interface for multiple clients (multitenancy), using centralized management, secured access and scalable cloud infrastructure.

Compute

The role and service of a Compute node in an OpenStack cluster consists in providing computing resources for running virtual instances by means of the Nova Compute component.

This node is responsible for VM life cycle management and the allocation of CPU, memory and disk space, and communicates with the control services based on requests from users or orchestration.

Compute apps

Represents a set of Compute applications. A set of Compute applications represents a group of software components that provide computing operations and management of computing resources within a cloud or virtualized infrastructure. It includes applications responsible for deploying, managing, scaling and monitoring virtual machines or containers, which work closely with the network, storage and orchestration layers of the system.

Control apps

The set of applications for the Control node represents the key software components that control and coordinate the operation of the entire cluster in an OpenStack or Kubernetes environment. This set includes services such as API servers, schedulers, state managers (controller managers), database services, authentication mechanisms and other system applications necessary for the orchestration, monitoring and life cycle management of workloads and infrastructure.

Compute

The role and service of a Compute node in an OpenStack cluster consists in providing computing resources for running virtual instances by means of the Nova Compute component. This node is responsible for VM life cycle management and the allocation of CPU, memory and disk space, and communicates with the control services based on requests from users or orchestration.

Compute apps

Represents a set of Compute applications. A set of Compute applications represents a group of software components that provide computing operations and management of computing resources within a cloud or virtualized infrastructure. It includes applications responsible for deploying, managing, scaling and monitoring virtual machines or containers, which work closely with the network, storage and orchestration layers of the system.

Compute

The role and service of a Compute node in an OpenStack cluster consists in providing computing resources for running virtual instances by means of the Nova Compute component. This node is responsible for VM life cycle management and the allocation of CPU, memory and disk space, and communicates with the control services based on requests from users or orchestration.

Compute apps

Represents a set of Compute applications. A set of Compute applications represents a group of software components that provide computing operations and management of

computing resources within a cloud or virtualized infrastructure. It includes applications responsible for deploying, managing, scaling and monitoring virtual machines or containers, which work closely with the network, storage and orchestration layers of the system.

Control apps

The set of applications for the Control node represents the key software components that control and coordinate the operation of the entire cluster in an OpenStack or Kubernetes environment. This set includes services such as API servers, schedulers, state managers (controller managers), database services, authentication mechanisms and other system applications necessary for the orchestration, monitoring and life cycle management of workloads and infrastructure.

Compute

The role and service of a Compute node in an OpenStack cluster consists in providing computing resources for running virtual instances by means of the Nova Compute component. This node is responsible for VM life cycle management and the allocation of CPU, memory and disk space, and communicates with the control services based on requests from users or orchestration.

Compute apps

Represents a set of Compute applications. A set of Compute applications represents a group of software components that provide computing operations and management of computing resources within a cloud or virtualized infrastructure. It includes applications responsible for deploying, managing, scaling and monitoring virtual machines or containers, which work closely with the network, storage and orchestration layers of the system.

Control apps

The set of applications for the Control node represents the key software components that control and coordinate the operation of the entire cluster in an OpenStack or Kubernetes environment. This set includes services such as API servers, schedulers, state managers (controller managers), database services, authentication mechanisms and other system applications necessary for the orchestration, monitoring and life cycle management of workloads and infrastructure.

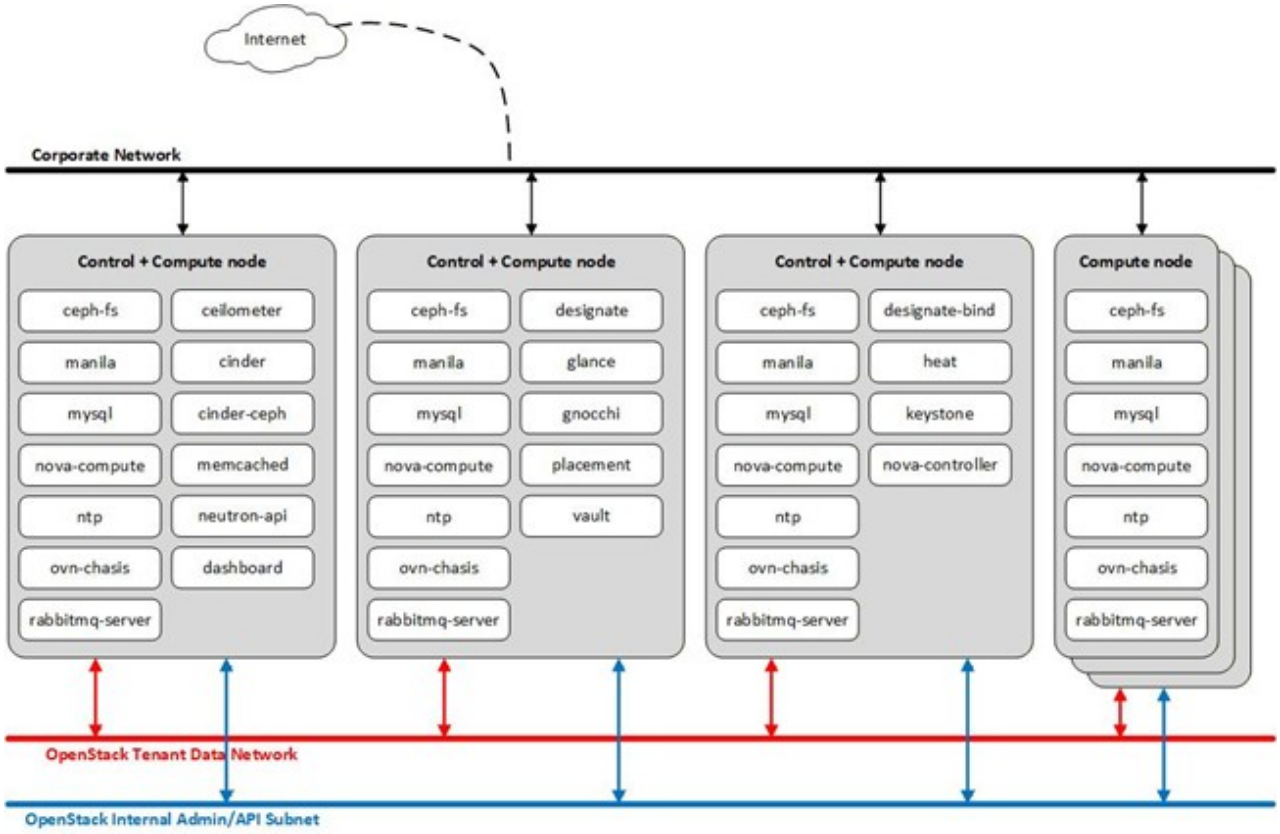
Compute

The role and service of a Compute node in an OpenStack cluster consists in providing computing resources for running virtual instances by means of the Nova Compute component. This node is responsible for VM life cycle management and the allocation of CPU, memory and disk space, and communicates with the control services based on requests from users or orchestration.

Compute apps

Represents a set of Compute applications. A set of Compute applications represents a group of software components that provide computing operations and management of computing resources within a cloud or virtualized infrastructure. It includes applications responsible for deploying, managing, scaling and monitoring virtual machines or containers, which work closely with the network, storage and orchestration layers of the system.

1.1.7. Technology Viewpoint OpenStack External



Obrázok1.png

Figure 17: Technology Viewpoint OpenStack External

1.1.8. Technology Viewpoint Reporting

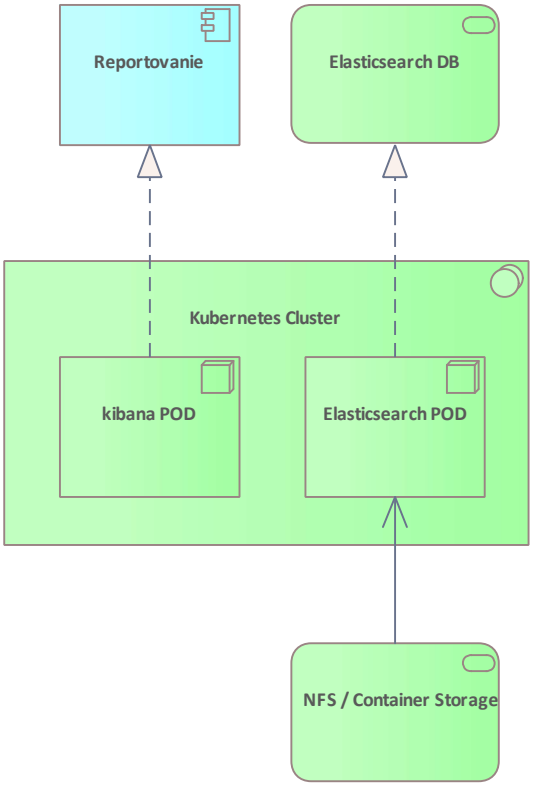


Figure 18: Technology Usage Viewpoint Reporting alt 1

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 1 represents the use of the Elasticsearch database directly within the module setup. The design envisages a containerized version of the NoSQL database for Elasticsearch.

Description of the elements used:

Reporting

The reporting component within the SOCaaS architecture serves as the central means for aggregating, visualizing and distributing security information. Its role is to provide both regular and ad hoc outputs on the state of cybersecurity for various target groups – from SOC analysts, through team leaders, to the management of the organization or clients. Such a reporting system must cover:

- Generation of metrics on incidents, alerts, SLAs and KPIs,
- Overviews of trends over time and evaluation of the performance of security measures,
- Adaptation of content according to the level of detail (technical vs. strategic),
- Automation and scheduling of the distribution of outputs,
- Retrospective analysis and archiving of historical data.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Elasticsearch DB

The technology service of the NoSQL Elasticsearch database is a distributed system for full-text search, indexing and data analysis that operates on the basis of a document-oriented model. It enables efficient processing of large amounts of unstructured or semi-structured data in real time, while being optimized for horizontal scaling, high availability and powerful analytics over logs, metrics or application data.

Elasticsearch POD

Represents the Elasticsearch POD for the NoSQL database, which is a container unit deployed in a Kubernetes cluster that hosts an Elasticsearch node as part of a distributed, document-oriented NoSQL database. This POD provides indexing, searching and storage of unstructured or semi-structured data, and is designed for horizontal scaling, high availability and fast response when working with large volumes of data.

Kibana POD

The Kibana POD for reporting is a container unit deployed in a Kubernetes cluster that hosts the Kibana web application intended for visualizing and analysing data stored in Elasticsearch. This POD provides a graphical user interface for creating dashboards, reports and interactive overviews, thereby enabling users to efficiently monitor, analyse and interpret operational and logging data in real time.

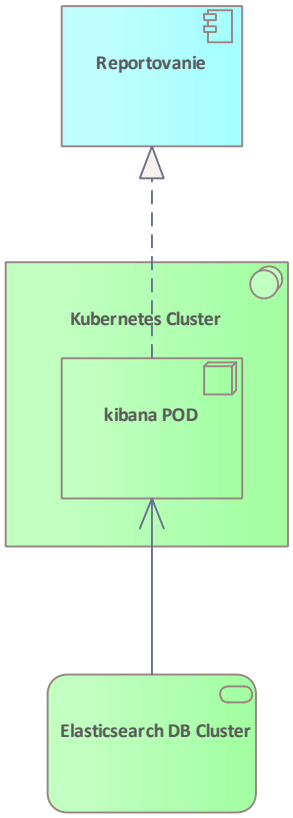


Figure 19: Technology Usage Viewpoint Reporting alt 2

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 2 represents the use of the Elasticsearch database directly in the form of a centralized DB Cluster.

Description of the elements used:

Reporting

The reporting component within the SOCaaS architecture serves as the central means for aggregating, visualizing and distributing security information. Its role is to provide both regular and ad hoc outputs on the state of cybersecurity for various target groups – from SOC analysts, through team leaders, to the management of the organization or clients. Such a reporting system must cover:

- Generation of metrics on incidents, alerts, SLAs and KPIs,
- Overviews of trends over time and evaluation of the performance of security measures,
- Adaptation of content according to the level of detail (technical vs. strategic),
- Automation and scheduling of the distribution of outputs,
- Retrospective analysis and archiving of historical data.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Elasticsearch DB Cluster

The technology service of the NoSQL Elasticsearch database in the form of a cluster represents a multi-node distributed solution in which the individual nodes cooperate on storing, indexing and searching document-oriented data. Each node in the cluster performs a specific role (e.g. master, data or ingest node), which enables horizontal scaling, high availability and fault tolerance, while the system as a whole ensures processing of large volumes of data in real time.

Kibana POD

The Kibana POD for reporting is a container unit deployed in a Kubernetes cluster that hosts the Kibana web application intended for visualizing and analysing data stored in Elasticsearch. This POD provides a graphical user interface for creating dashboards, reports and interactive overviews, thereby enabling users to efficiently monitor, analyse and interpret operational and logging data in real time.

1.1.9. Technology Viewpoint SIEM

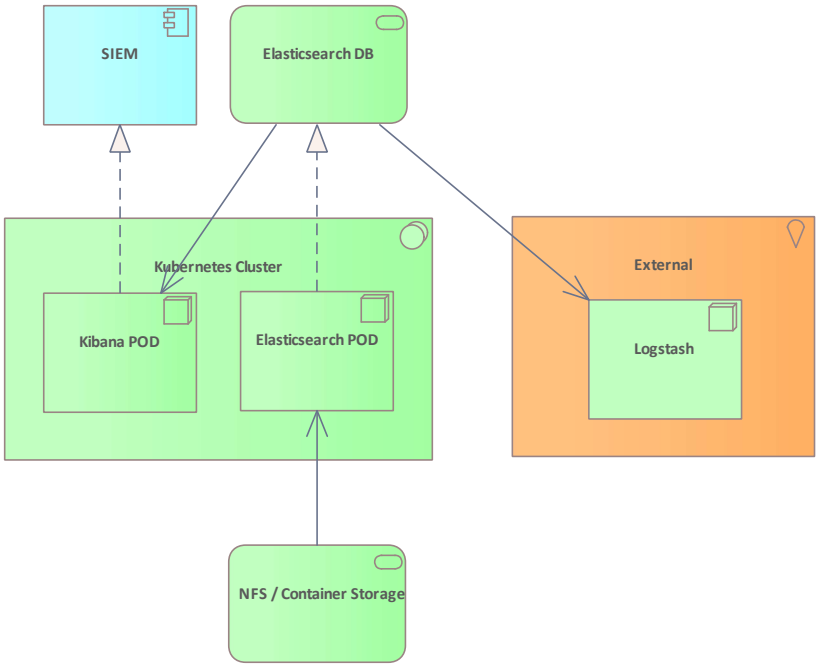


Figure 20: Technology Usage Viewpoint SIEM alt 1

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 1 represents the use of the Elasticsearch database directly within the module setup. The design envisages a containerized version of NoSQL databases for Elasticsearch.

Description of the elements used:

SIEM

SIEM is the heart of the SOCaaS solution. It provides aggregation, processing, correlation and analysis of security events from various sources in real time. It enables analysts to identify anomalies, generate alerts and escalate incidents, while increasing visibility over the client's infrastructure.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Elasticsearch DB

The technology service of the NoSQL Elasticsearch database is a high-performance, distributed system designed for indexing, searching and analysing data in real time, based on a document-oriented model. It uses a JSON document schema, supports horizontal scaling and high availability, and is optimized for processing large volumes of unstructured or semi-structured data such as logs, metrics or application records.

Elasticsearch POD

The Elasticsearch POD for the NoSQL database is a container instance deployed in a Kubernetes environment that hosts an Elasticsearch node as part of a distributed NoSQL database cluster. This POD provides indexing, searching and storage of document-oriented data, while enabling horizontal scaling, high availability and efficient processing of large volumes of unstructured information in real time.

External

Represents the physical location at the client and thus outside the INNOSOC environment.

Kibana POD

The Kibana POD for SIEM is a container instance deployed in a Kubernetes cluster that provides a graphical user interface for visualizing and analysing security data collected in Elasticsearch. It serves as a frontend component for the SIEM (Security Information and Event Management) solution, enabling interactive monitoring, event correlation, creation of dashboards and detection of security incidents in real time.

Logstash

A Logstash instance for log processing is a data processor deployed as a standalone service or container that collects, transforms and routes logging data from various sources. Within the pipeline, it provides parsing, enrichment and filtering of data, which it then writes to Elasticsearch, thereby enabling centralized, efficient and structured storage of logs for further analysis.

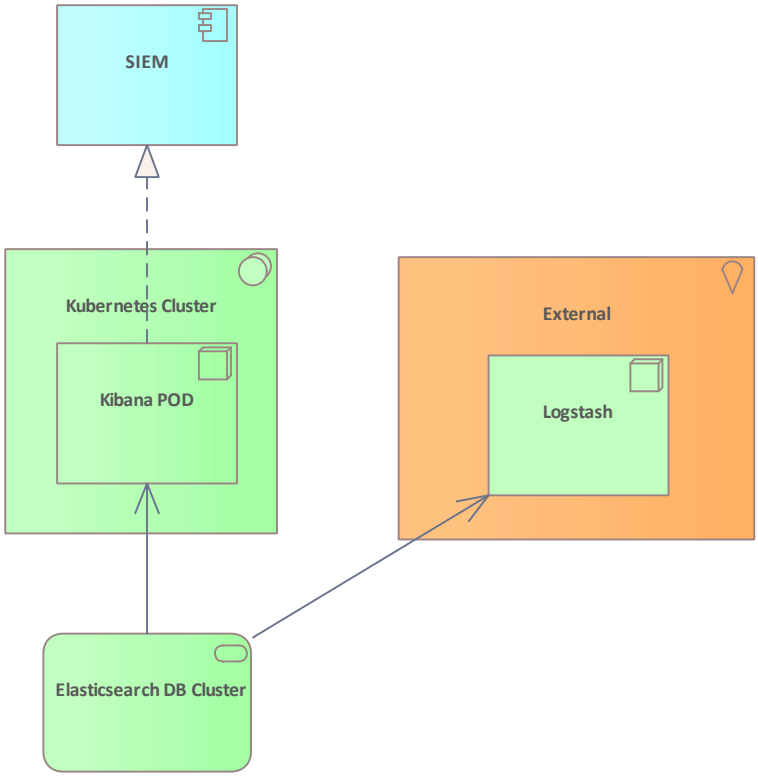


Figure 21: Technology Usage Viewpoint SIEM alt 2

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 2 represents the use of the Elasticsearch database directly in the form of a centralized DB Cluster.

Description of the elements used:

SIEM

SIEM is the heart of the SOCaaS solution. It provides aggregation, processing, correlation and analysis of security events from various sources in real time. It enables analysts to identify anomalies, generate alerts and escalate incidents, while increasing visibility over the client's infrastructure.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Elasticsearch DB Cluster

The technology service of the NoSQL Elasticsearch database in the form of a cluster represents a multi-node distributed solution in which the individual nodes cooperate on storing, indexing and searching document-oriented data. Each node in the cluster performs a specific role (e.g. master, data or ingest node), which enables horizontal scaling, high availability and fault tolerance, while the system as a whole ensures processing of large volumes of data in real time.

External

Represents the physical location at the client and thus outside the INNOSOC environment.

Kibana POD

The Kibana POD for SIEM is a container instance deployed in a Kubernetes cluster that provides a graphical user interface for visualizing and analysing security data collected in Elasticsearch. It serves as a frontend component for the SIEM (Security Information and Event Management) solution, enabling interactive monitoring, event correlation, creation of dashboards and detection of security incidents in real time.

Logstash

A Logstash instance for log processing is a data processor deployed as a standalone service or container that collects, transforms and routes logging data from various sources. Within the pipeline, it provides parsing, enrichment and filtering of data, which it then writes to Elasticsearch, thereby enabling centralized, efficient and structured storage of logs for further analysis.

1.1.10. Technology Viewpoint Vulnerability Scanner

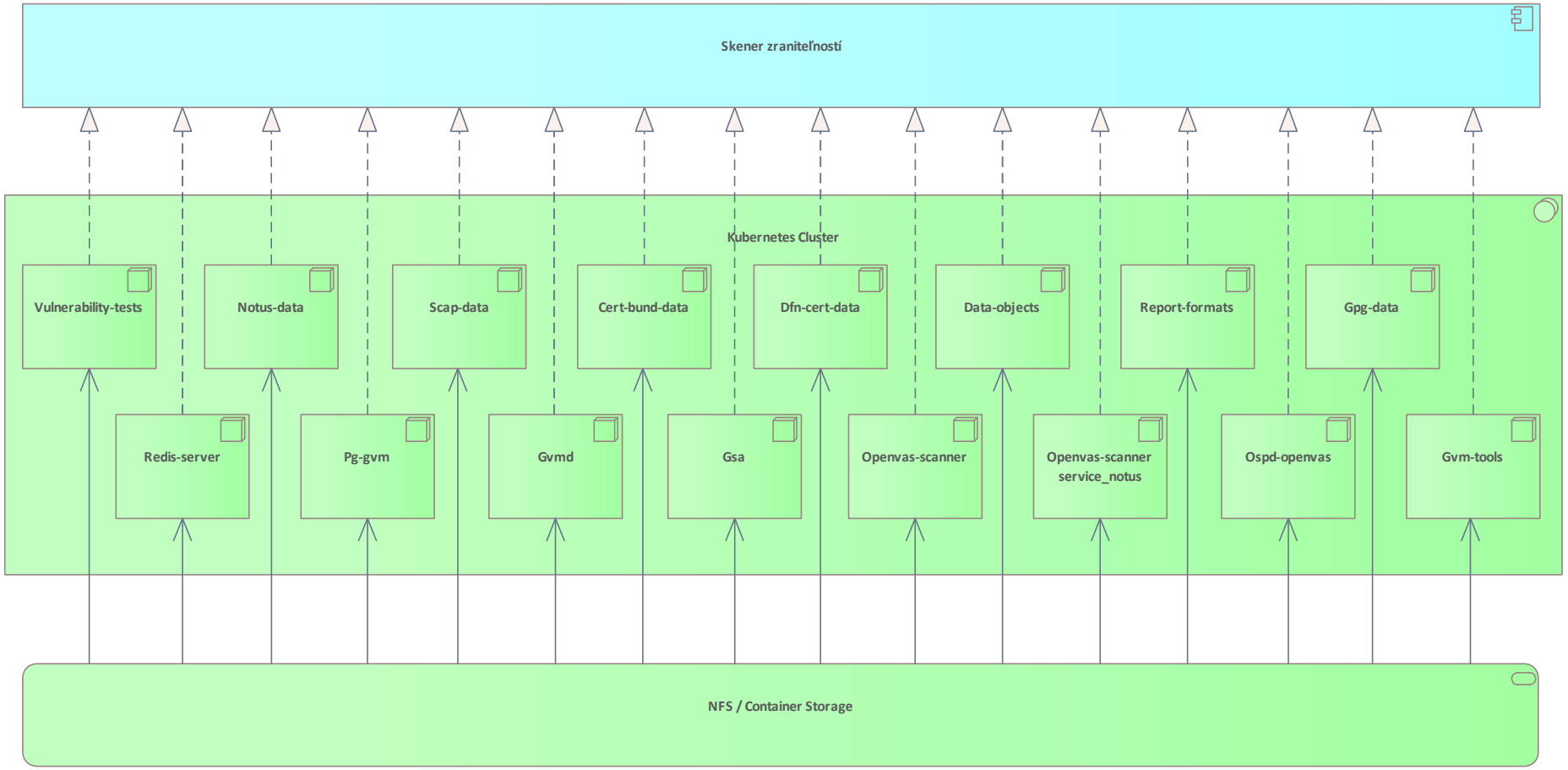


Figure 22: Technology Usage Viewpoint Vulnerability Scanner

The diagram shows a simplified assignment of the individual PODs to the corresponding application components.

Description of the elements used:

Vulnerability scanner

The vulnerability scanner is a fundamental building block for ensuring a proactive approach in the SOCaaS architecture. It makes it possible to regularly identify weaknesses in the infrastructure, assess the status of updates, configurations and settings, and provide information for further prioritization and planning of corrective measures. The scanner outputs are often a source for SOAR systems, SLA metrics and reporting, as well as for assessing the client's security level.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Cert-bund-data

The cert-bund-data POD is a container unit deployed in a Kubernetes cluster that provides updated security information and advisories from official sources. This POD distributes data on threats, vulnerabilities and recommendations, which are used by security tools to improve the detection of and response to current cyber risks.

Data-objects

The data-objects POD is a container unit deployed in a Kubernetes cluster that serves to store and provide structured data objects used within the security ecosystem, such as definitions of vulnerabilities, configurations or analytical rules. This POD provides centralized access to reference data for other system components, thereby supporting the consistency and currency of information in the processes of detecting and assessing security threats.

Dfn-cert-data

The dfn-cert-data POD is a container unit deployed in a Kubernetes cluster that serves to distribute security information and advisories. This POD provides up-to-date data on cyber threats, vulnerabilities and recommendations, which are used by security tools to improve detection capabilities and enable faster incident response.

Gpg-data

The gpg-data POD is a container unit deployed in a Kubernetes cluster that provides GPG (GNU Privacy Guard) keys and related data necessary for verifying the signatures of update packages and security data in the Greenbone Vulnerability Management (GVM) ecosystem. This POD ensures the integrity and trustworthiness of the transmitted security information, such as vulnerability databases, report formats or configuration files.

Gsa

The gsa POD is a container unit deployed in a Kubernetes cluster that hosts the Greenbone Security Assistant (GSA) component – a web user interface for working with the Greenbone Vulnerability Management (GVM) system. This POD provides access to vulnerability scanning functions via a browser, including user management, scan scheduling, display of results and configuration of the entire security environment.

Gvmd

The gvmd POD is a container unit deployed in a Kubernetes cluster that hosts the Greenbone Vulnerability Management Daemon (gvmd) component – the central control service for the GVM/OpenVAS tool. This POD provides user management, scheduling and launching of scans, processing of results and communication with the database and other services, thereby forming the core of the vulnerability scanning system.

Gvm-tools

The gvm-tools POD is a container unit deployed in a Kubernetes cluster that contains the GVM-Tools – client utilities for automated management of and interaction with the Greenbone Vulnerability Management (GVM) system. This POD enables the execution of commands, scripting of tasks, scheduling of scans and extraction of results via XML or GMP (Greenbone Management Protocol), thereby supporting the integration of GVM into larger security and orchestration workflows.

Notus-data

The notus-data POD is a container unit deployed in a Kubernetes cluster that provides data and updates for vulnerability detection mechanisms, particularly in the context of the Greenbone/OpenVAS tool. This POD distributes structured information on known vulnerabilities in JSON format, which is used by other components for precise and up-to-date testing of systems for security weaknesses.

Openvas-scanner

The openvas-scanner POD is a container unit deployed in a Kubernetes cluster that hosts the OpenVAS (Open Vulnerability Assessment System) component – a powerful

scanning engine for vulnerability testing. This POD performs active scanning of target systems, analyses their security weaknesses based on updated signatures and communicates with the central gvmd daemon to record and process the scan results.

Openvas-scanner service_notus

Represents the openvas-scanner service_notus POD. The openvas-scanner service_notus POD is a container unit deployed in a Kubernetes cluster that extends the functionality of the OpenVAS scanning engine with integration with the Notus service. This service enables agentless and fast vulnerability testing based on data from the Notus database without the need for direct network access to all targets. The POD combines the performance of active scanning with dynamic detection of known vulnerabilities, thereby increasing the accuracy and efficiency of security analyses.notus.

Ospd-openvas

The ospd-openvas POD is a container unit deployed in a Kubernetes cluster that hosts the OSPd (Open Scanner Protocol daemon) component for integrating the OpenVAS scanner with the gvmd control system within Greenbone Vulnerability Management. This POD ensures standardized communication between gvmd and OpenVAS, processes scan requests, controls their execution and sends the results back for central processing, thereby enabling scalable and efficient orchestration of vulnerability scans.

Pg-gvm

The pg-gvm POD is a container unit deployed in a Kubernetes cluster that hosts a PostgreSQL database adapted for Greenbone Vulnerability Management (GVM). This POD serves as the central store for all data on vulnerabilities, scans, results and configurations that are generated and managed by the GVM/OpenVAS tools within the security ecosystem.

Redis-server

The redis-server POD is a container unit deployed in a Kubernetes cluster that hosts a Redis instance – a high-performance in-memory NoSQL database. This POD provides temporarily or persistently stored key-value data for fast access, caching, message queuing or sharing state between applications within a distributed system.

Report-formats

Represents the report-formats POD. The report-formats POD is a container unit deployed in a Kubernetes cluster that provides and manages templates and formats of output reports within the Greenbone Vulnerability Management (GVM) system. This POD enables scan results to be generated in various standardized formats, thereby supporting automated reporting, archiving and sharing of findings with various stakeholders.

Scap-data

The scap-data POD is a container unit deployed in a Kubernetes cluster that provides structured security data packages in SCAP (Security Content Automation Protocol) format. These data contain standardized information on security configurations, vulnerabilities and

policies, which are subsequently used by tools for compliance audits, conformity with security standards and automated assessment of systems.

Vulnerability-tests

The vulnerability-tests POD is a container unit deployed in a Kubernetes cluster that performs automated vulnerability tests against target systems, services or containers. This POD uses security scanners or analysis tools and serves to detect known weaknesses, assess risks and generate outputs for further processing within the security cycle.

1.1.11. Technology Viewpoint SOAR

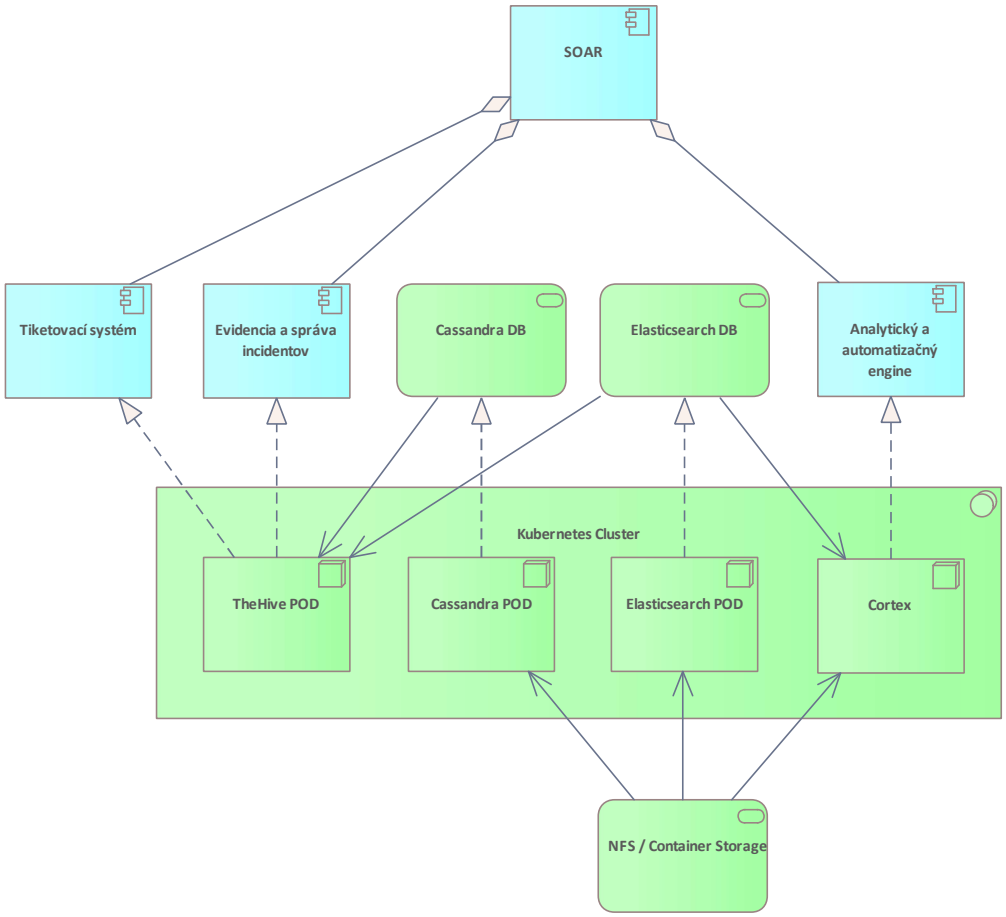


Figure 23: Technology Usage Viewpoint SOAR alt 1

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 1 represents the use of the Elasticsearch database directly within the module setup. The design envisages a containerized version of NoSQL databases for Cassandra and Elasticsearch.

Description of the elements used:

Analytics and automation engine

The analytics and automation engine is a software component or service that combines data analysis and the automation of decision-making or operational processes with the aim of increasing the efficiency, accuracy and speed of the system's response. In the context of cybersecurity or IT operations, this engine processes input data (e.g. logs, events, metrics), applies analytical rules or models (e.g. anomaly detection, correlation), and subsequently performs automated responses or actions through integrated tools and API interfaces.

Incident recording and management

Incident recording and management is a core element of every security operation (SOC). This subcomponent ensures systematic documentation, tracking and management of security events and incidents throughout their life cycle – from detection through analysis to closure. A high-quality recording system must enable teams to collaborate in a coordinated manner, document steps, record related artefacts (observables) and link incidents

with relevant threats, systems or historical cases. Ideally, it also supports advanced data enrichment, auditing, multi-tenancy and the possibility of retrospective investigation of incidents.

SOAR

SOAR (Security Orchestration, Automation and Response) is the component responsible for automating incident responses, orchestrating security tools and supporting analysts' decision-making. In SOCaaS, it is a fundamental element enabling the scaling of response capacities without the need to increase staff numbers. It is indispensable especially when processing large volumes of alerts, where it reduces the MTTR and MTTC metrics.

Ticketing system

The ticketing system serves as the central communication and organizational channel for the management of security incidents throughout their entire life cycle. It enables the creation, assignment and tracking of tasks and communication between analysts, while supporting audit, SLA monitoring, escalation and notifications.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes

cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Cassandra DB

The technology service of the NoSQL Cassandra database is a distributed database system designed for storing large volumes of structured data with an emphasis on high availability, scalability and fault tolerance. It uses a masterless architecture with data replication across nodes, which enables fast writing and reading in a geographically distributed environment without a single point of failure.

Elasticsearch DB

The technology service of the NoSQL Elasticsearch database is a high-performance, distributed system designed for indexing, searching and analysing data in real time, based on a document-oriented model. It uses a JSON document schema, supports horizontal scaling and high availability, and is optimized for processing large volumes of unstructured or semi-structured data such as logs, metrics or application records.

Cassandra POD

The Apache Cassandra POD for the NoSQL database is a container instance deployed in a Kubernetes environment that hosts a Cassandra node as part of a distributed NoSQL cluster. This POD provides scalable, highly available and resilient storage of structured data with an emphasis on fast writing and reading in an environment with large volumes of data and high redundancy.

Elasticsearch POD

The Elasticsearch POD for the NoSQL database is a container instance deployed in a Kubernetes environment that hosts an Elasticsearch node as part of a distributed NoSQL database cluster. This POD provides indexing, searching and storage of document-oriented data, while enabling horizontal scaling, high availability and efficient processing of large volumes of unstructured information in real time.

TheHive POD

The TheHive POD for asset inventory is a container instance deployed in a Kubernetes environment that hosts the TheHive application – an open-source platform for managing security incidents and digital assets. This POD enables centralized collection, categorization and analysis of assets, their linking to incidents and support for collaborative investigation within cybersecurity teams.

Cortex

Represents the Cortex POD for SOAR. The Cortex POD for SOAR is a container unit deployed in a Kubernetes environment that hosts the Cortex service as part of the SOAR solution. This POD enables automated launching of analytical modules (analyzers) and

response actions (responders) based on security incidents, thereby streamlining threat detection, incident response and integration with other security tools.

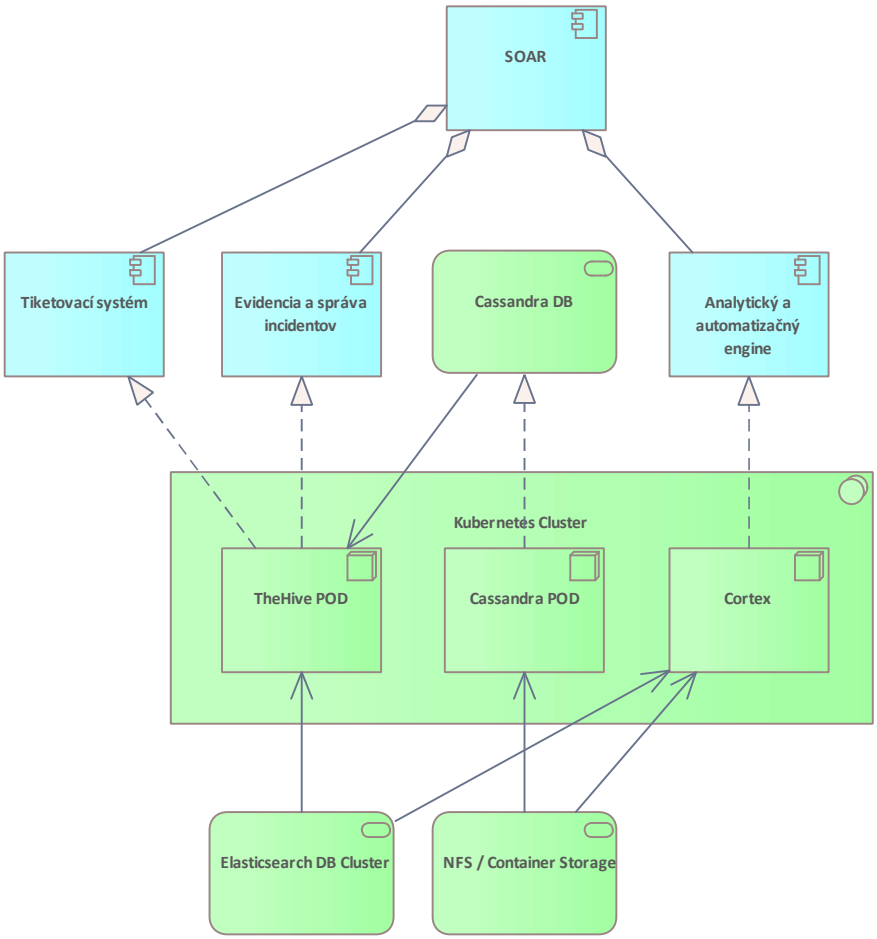


Figure 24: Technology Usage Viewpoint SOAR alt 2

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 2 represents the use of the Elasticsearch database directly in the form of a centralized DB Cluster. The design envisages a containerized version of the NoSQL database for Cassandra.

Description of the elements used:

Analytics and automation engine

The analytics and automation engine is a software component or service that combines data analysis and the automation of decision-making or operational processes with the aim of increasing the efficiency, accuracy and speed of the system's response. In the context of cybersecurity or IT operations, this engine processes input data (e.g. logs, events, metrics), applies analytical rules or models (e.g. anomaly detection, correlation), and subsequently performs automated responses or actions through integrated tools and API interfaces.

Incident recording and management

Incident recording and management is a core element of every security operation (SOC). This subcomponent ensures systematic documentation, tracking and management of security events and incidents throughout their life cycle – from detection through analysis to closure. A high-quality recording system must enable teams to collaborate in a coordinated manner, document steps, record related artefacts (observables) and link incidents

with relevant threats, systems or historical cases. Ideally, it also supports advanced data enrichment, auditing, multi-tenancy and the possibility of retrospective investigation of incidents.

SOAR

SOAR (Security Orchestration, Automation and Response) is the component responsible for automating incident responses, orchestrating security tools and supporting analysts' decision-making. In SOCaaS, it is a fundamental element enabling the scaling of response capacities without the need to increase staff numbers. It is indispensable especially when processing large volumes of alerts, where it reduces the MTTR and MTTC metrics.

Ticketing system

The ticketing system serves as the central communication and organizational channel for the management of security incidents throughout their entire life cycle. It enables the creation, assignment and tracking of tasks and communication between analysts, while supporting audit, SLA monitoring, escalation and notifications.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Cassandra DB

The technology service of the NoSQL Cassandra database is a distributed database system designed for storing large volumes of structured data with an emphasis on high availability, scalability and fault tolerance. It uses a masterless architecture with data replication

across nodes, which enables fast writing and reading in a geographically distributed environment without a single point of failure.

Elasticsearch DB Cluster

The technology service of the NoSQL Elasticsearch database in the form of a cluster represents a multi-node distributed solution in which the individual nodes cooperate on storing, indexing and searching document-oriented data. Each node in the cluster performs a specific role (e.g. master, data or ingest node), which enables horizontal scaling, high availability and fault tolerance, while the system as a whole ensures processing of large volumes of data in real time.

Cassandra POD

The Apache Cassandra POD for the NoSQL database is a container instance deployed in a Kubernetes environment that hosts a Cassandra node as part of a distributed NoSQL cluster. This POD provides scalable, highly available and resilient storage of structured data with an emphasis on fast writing and reading in an environment with large volumes of data and high redundancy.

TheHive POD

The TheHive POD for asset inventory is a container instance deployed in a Kubernetes environment that hosts the TheHive application – an open-source platform for managing security incidents and digital assets. This POD enables centralized collection, categorization and analysis of assets, their linking to incidents and support for collaborative investigation within cybersecurity teams.

Cortex

Represents the Cortex POD for SOAR. The Cortex POD for SOAR is a container unit deployed in a Kubernetes environment that hosts the Cortex service as part of the SOAR solution. This POD enables automated launching of analytical modules (analyzers) and response actions (responders) based on security incidents, thereby streamlining threat detection, incident response and integration with other security tools.

1.1.12. Technology Viewpoint Ticketing System / Incident Recording and Management

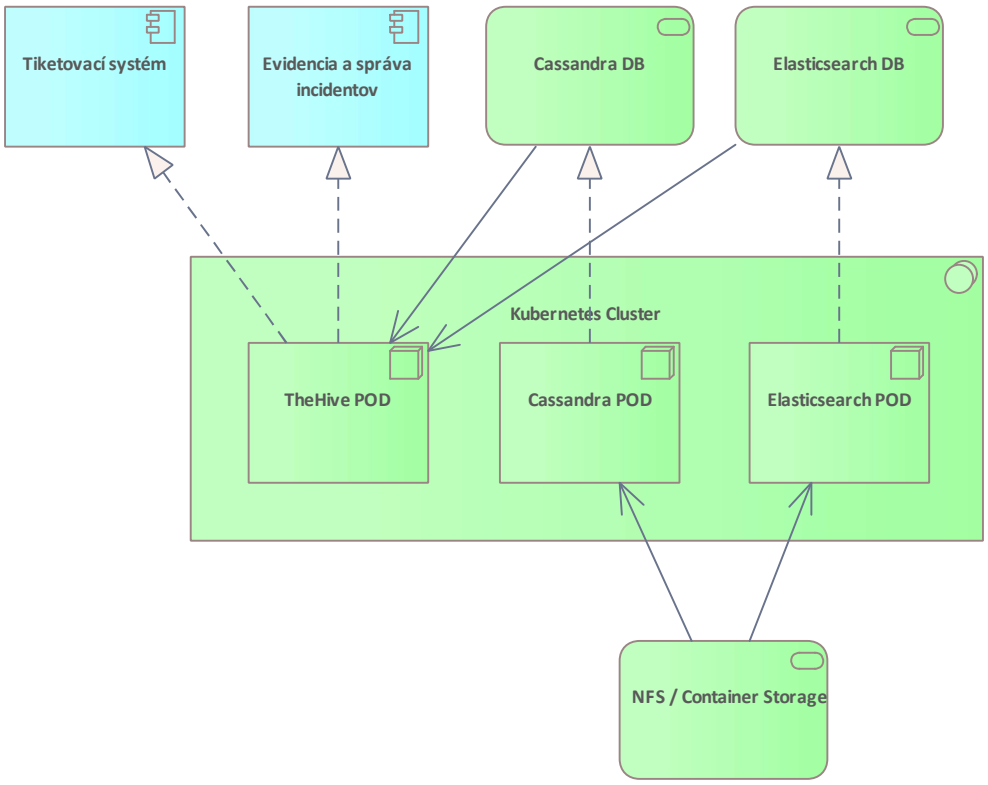


Figure 25: Technology Usage Viewpoint Ticketing System/
Incident Recording and Management alt 1

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 1 represents the use of the Elasticsearch database directly within the module setup. The design envisages a containerized version of NoSQL databases for Cassandra and Elasticsearch.

Description of the elements used:

Incident recording and management

Incident recording and management is a core element of every security operation (SOC). This subcomponent ensures the systematic documentation, tracking and management of security events and incidents throughout their life cycle – from detection through analysis to closure. A good-quality recording system must enable teams to collaborate in a coordinated manner, document steps, record related artefacts (observables) and link incidents with relevant threats, systems or historical cases. Ideally, it also supports advanced data enrichment, auditing, multi-tenancy and the possibility of retrospective investigation of incidents.

Ticketing system

The ticketing system serves as the central communication and organizational channel for the management of security incidents throughout their entire life cycle. It enables the

creation, assignment and tracking of tasks and communication between analysts, while supporting audit, SLA monitoring, escalation and notifications.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Cassandra DB

The technology service of the NoSQL Cassandra database is a distributed database system designed for storing large volumes of structured data with an emphasis on high availability, scalability and fault tolerance. It uses a masterless architecture with data replication across nodes, which enables fast writing and reading in a geographically distributed environment without a single point of failure.

Elasticsearch DB

The technology service of the NoSQL Elasticsearch database is a high-performance, distributed system designed for indexing, searching and analysing data in real time, based on a document-oriented model. It uses a JSON document schema, supports horizontal scaling and high availability, and is optimized for processing large volumes of unstructured or semi-structured data such as logs, metrics or application records.

Cassandra POD

The Apache Cassandra POD for the NoSQL database is a container instance deployed in a Kubernetes environment that hosts a Cassandra node as part of a distributed NoSQL cluster. This POD provides scalable, highly available and resilient storage of structured data with an emphasis on fast writing and reading in an environment with large volumes of data and high redundancy.

Elasticsearch POD

The Elasticsearch POD for the NoSQL database is a container instance deployed in a Kubernetes environment that hosts an Elasticsearch node as part of a distributed NoSQL

database cluster. This POD provides indexing, searching and storage of document-oriented data, while enabling horizontal scaling, high availability and efficient processing of large volumes of unstructured information in real time.

TheHive POD

The TheHive POD for asset inventory is a container instance deployed in a Kubernetes environment that hosts the TheHive application – an open-source platform for managing security incidents and digital assets. This POD enables centralized collection, categorization and analysis of assets, their linking to incidents and support for collaborative investigation within cybersecurity teams.

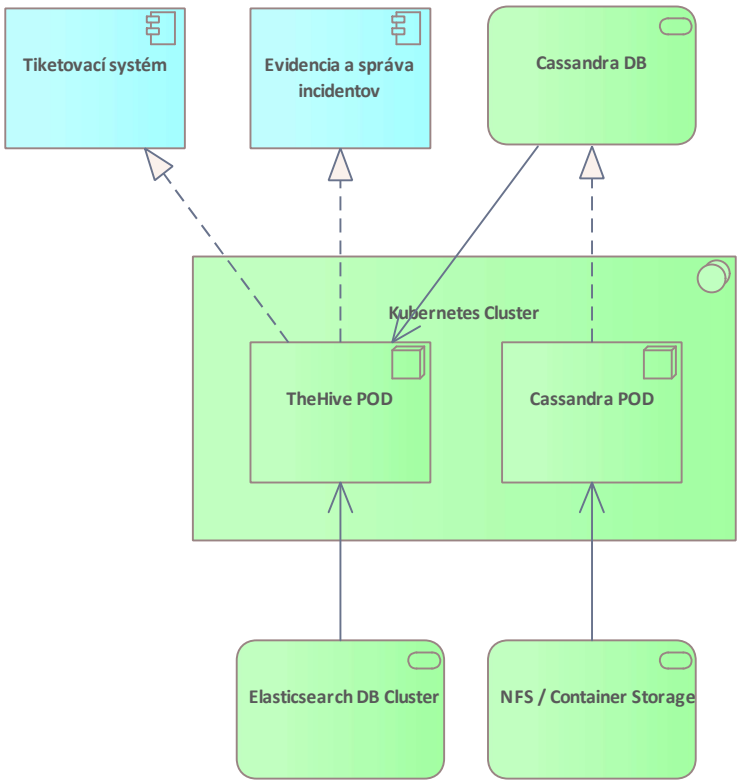


Figure 26: Technology Usage Viewpoint Ticketing System/
Incident Recording and Management alt 2

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 2 represents the use of the Elasticsearch database directly in the form of a centralized DB Cluster. The design envisages a containerized version of the NoSQL database for Cassandra.

Description of the elements used:

Incident recording and management

Incident recording and management is a core element of every security operation (SOC). This subcomponent ensures systematic documentation, tracking and management of security events and incidents throughout their life cycle – from detection through analysis to

closure. A high-quality recording system must enable teams to collaborate in a coordinated manner, document steps, record related artefacts (observables) and link incidents

with relevant threats, systems or historical cases. Ideally, it also supports advanced data enrichment, auditing, multi-tenancy and the possibility of retrospective investigation of incidents.

Ticketing system

The ticketing system serves as the central communication and organizational channel for the management of security incidents throughout their entire life cycle. It enables the creation, assignment and tracking of tasks and communication between analysts, while supporting audit, SLA monitoring, escalation and notifications.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Cassandra DB

The technology service of the NoSQL Cassandra database is a distributed database system designed for storing large volumes of structured data with an emphasis on high availability, scalability and fault tolerance. It uses a masterless architecture with data replication across nodes, which enables fast writing and reading in a geographically distributed environment without a single point of failure.

Elasticsearch DB Cluster

The technology service of the NoSQL Elasticsearch database in the form of a cluster represents a multi-node distributed solution in which the individual nodes cooperate on storing, indexing and searching document-oriented data. Each node in the cluster performs a specific role (e.g. master, data or ingest node), which enables horizontal scaling, high availability and fault tolerance, while the system as a whole ensures processing of large volumes of data in real time.

Cassandra POD

The Apache Cassandra POD for the NoSQL database is a container instance deployed in a Kubernetes environment that hosts a Cassandra node as part of a distributed NoSQL cluster. This POD provides scalable, highly available and resilient storage of structured data with an emphasis on fast writing and reading in an environment with large volumes of data and high redundancy.

TheHive POD

The TheHive POD for asset inventory is a container instance deployed in a Kubernetes environment that hosts the TheHive application – an open-source platform for managing security incidents and digital assets. This POD enables centralized collection, categorization and analysis of assets, their linking to incidents and support for collaborative investigation within cybersecurity teams.

1.1.13. Technology Viewpoint Knowledge Portal for the SOC Team

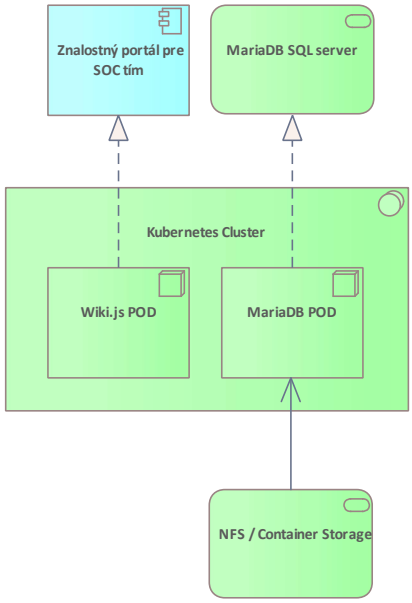


Figure 27: Technology Usage Viewpoint Knowledge Portal for the SOC Team alt 1

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 1 represents the use of the database directly within the module setup. The design envisages a containerized version of the SQL server.

Description of the elements used:

Knowledge portal for the SOC team

A knowledge portal for the SOC team is an important building block of every modern security operations centre. It serves as a central repository of know-how, playbooks, procedures, recommendations, analytical guides and often also historical lessons learned from incidents. The knowledge portal:

- Increases the efficiency and consistency of incident response,
- Enables rapid onboarding of new team members,
- Supports the standardization of processes and methodologies,
- Enables the team to retain and share proven procedures and its own analytical experience,
- Strengthens the ability to learn collectively from incidents and to build internal "threat intel" knowledge.

NFS/Container Storage

Represents the NFS service or other Container Storage. NFS (Network File System) is a network file protocol of the application layer of the OSI model that enables remote access to file systems over a network. It is one of the most common forms of shared storage in containerized environments such as Kubernetes, where it is implemented as a PersistentVolume (PV) type. In a broader sense, container storage refers to the mechanisms and services providing storage space for containers that need persistent or shared data between restarts or between cluster nodes.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

MariaDB SQL server

The technology service of the MariaDB relational database is a powerful and scalable relational database system built on the SQL architecture, which serves for persistent storage, management and querying of structured data using relational tables. MariaDB supports transactions, referential integrity, replication and high availability, which makes it suitable for deployment in critical enterprise applications where performance, reliability and security of data processing are important.

MariaDB POD

The MariaDB POD for the relational SQL database is a container unit deployed in a Kubernetes cluster that hosts the MariaDB database server – a powerful open-source relational database system compatible with MySQL. This POD provides persistently stored, structured data for applications or services, supports transactions and data integrity, and enables horizontal or vertical scaling within the cloud infrastructure.

Wiki.js POD

The Wiki.js POD for the Knowledge Portal for the SOC Team is a container unit deployed in a Kubernetes cluster that hosts the Wiki.js application – a modern and powerful system for managing documentation and knowledge. This POD serves as the central knowledge portal for the cybersecurity (SOC) team, where information on procedures,

incidents, analyses and internal processes is stored, managed and shared with an emphasis on security, auditability and access rights.

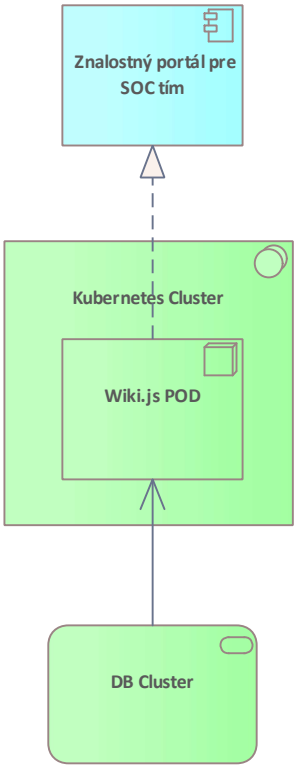


Figure 28: Technology Usage Viewpoint Knowledge Portal for the SOC Team alt 2

The diagram shows a simplified assignment of the individual PODs to the corresponding application components. Alt 2 represents the use of the database directly in the form of a centralized DB Cluster.

Description of the elements used:

Knowledge portal for the SOC team

A knowledge portal for the SOC team is an important building block of every modern security operations centre. It serves as a central repository of know-how, playbooks, procedures, recommendations, analytical guides and often also historical lessons learned from incidents. The knowledge portal:

- Increases the efficiency and consistency of incident response,
- Enables rapid onboarding of new team members,
- Supports the standardization of processes and methodologies,
- Enables the team to retain and share proven procedures and its own analytical experience,
- Strengthens the ability to learn collectively from incidents and to build internal "threat intel" knowledge.

DB Cluster

Represents a clustered relational database service. A clustered relational database service is a distributed database system that provides relational database services through multiple interconnected database nodes (clusters), thereby ensuring high availability, scalability, redundancy and fault tolerance.

Kubernetes Cluster

Represents a Kubernetes Cluster. A Kubernetes Cluster is a distributed computing environment used for the orchestration of containerized applications. It provides deployment, scaling, monitoring, updating and management of containers (most commonly Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated application life cycle management.

Wiki.js POD

The Wiki.js POD for the Knowledge Portal for the SOC Team is a container unit deployed in a Kubernetes cluster that hosts the Wiki.js application – a modern and powerful system for managing documentation and knowledge. This POD serves as the central knowledge portal for the cybersecurity (SOC) team, where information on procedures, incidents, analyses and internal processes is stored, managed and shared with an emphasis on security, auditability and access rights.