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
1.1. Operating environments 3
2. technological architecture 3
1.1. Technology Viewpoint CTI 4
1.2. Technology Viewpoint EDR/XDR 11
1.3. Technology Viewpoint INNOSOC Common 16
1.4. Technology Viewpoint Asset inventory 19
1.5. Technology Viewpoint Kubernetes 23
1.6. Technology Viewpoint OpenStack 48
1.7. Technology Viewpoint Reporting 52
1.8. Technology Viewpoint SIEM 56
1.9. Technology Viewpoint Vulnerability scanner 60
1.10. Technology Viewpoint SOAR 63
1.11. Technology Viewpoint Ticketing system / Incident recording and management 69
1.12. Technology Viewpoint Knowledge portal for the SOC team 74
3. Conclusion 78

1. INTRODUCTION

The aim of the project is to develop a technologically modern, open and scalable solution that will be able to support organizations in effectively handling cyber threats and ensuring operational resilience. The architecture of the solution is designed to meet the principles of high availability, security and flexibility, while taking into account current standards and best practice in the field of information technology and cybersecurity.

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

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.

1.1. Operating environments

The proposed operation is planned in two identical environments, which differ only in the quantity and type of resources:

PROD environment – production environment with real data and operation of the public portal.

TEST environment – test, training and pre-production environment. The entire TEST environment is accessible via VPN for privileged users.

2. TECHNOLOGICAL ARCHITECTURE

Technological architecture represents one of the key layers of the organization's architectural framework, focused on the design, selection, integration and management of the infrastructure and platform components that support the running of applications, services and processes. It serves as a link between business needs and the technical solution, defining how the technological resources and tools will be used to achieve the required functional and non-functional objectives of the system**.** It includes the hardware environment, virtualization, network layers, containerization platforms, CI/CD pipeline management, security mechanisms and the method of deployment, operation and monitoring of both system and application components.

In the context of modern information systems – especially those operated as multi-tenant cloud services – technological architecture is essential for ensuring scalability, high availability, security and efficient operation. Its design must reflect not only current requirements but also the system's ability to respond to future changes, support continuous development (DevOps/DevSecOps), integrate automation and enable flexible expansion without major interventions in the existing operation.

Technological architecture also includes the principles for designing deployment units (e.g. PODs for the frontend, backend and supporting services), the use of container technologies and orchestration using Kubernetes, deployment via GitLab CI/CD Workers, storing artefacts in the Docker Registry, and securing communication using valid SSL certificates and reverse proxy layers (e.g. HAProxy). It also incorporates mechanisms for managing network subnets, secured access to system services and the integration of monitoring and logging tools.

Technological architecture thus creates the technical foundation of the entire solution – it guarantees consistent, automated and secure deployment of services, efficient operations management and the ability to respond dynamically to changing environmental conditions as well as to user requirements.

1.1. Technology Viewpoint CTI

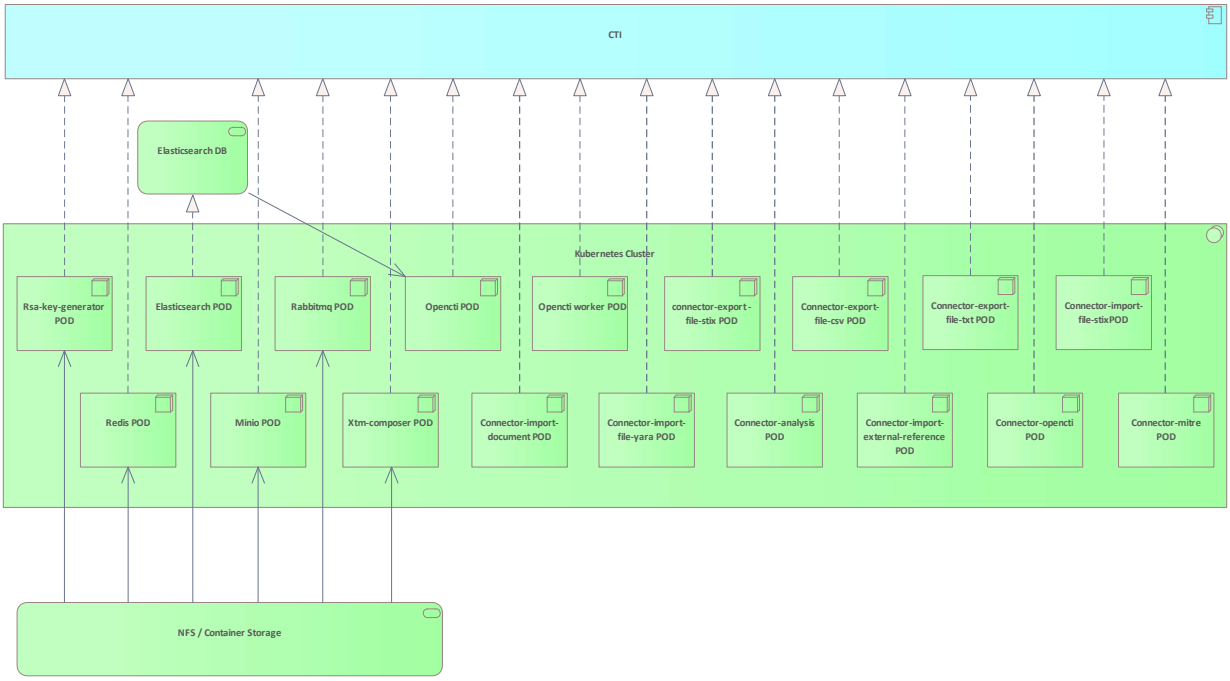


Figure 1 : Technology Usage Viewpoint CTI alt 1

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 the 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 capability to enrich detection rules, correlation mechanisms and playbooks with external threat information – including IOCs, TTPs, attacker profiling, campaigns and related vulnerabilities. In the environment of the multi-customer SOCaaS service, 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,
- Support of threat hunting and forensic analysis processes,
- Prioritization of incidents according to 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 ensures the deployment, scaling, monitoring, updating and management of containers (most often Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, the computing resources (CPU, memory, storage, network) are

abstracted and managed as a single system, which enables automated management of the application life cycle.

Elasticsearch DB

Represents the technology service of the NoSQL Elasticsearch database.

Elasticsearch POD

Represents the Elasticsearch POD for the NoSQL database.

Connector-analysis POD

Represents the connector-analysis POD.

Connector-export- file-csv POD

Represents the connector-export-file-csv POD.

connector-export -file-stix POD

Represents the connector-export-file-stix POD.

Connector-export- file-txt POD

Represents the connector-export-file-txt POD.

Connector-import- document POD

Represents the connector-import-document POD.

Connector-import- external-reference POD

Represents the connector-import-external-reference POD.

Connector-import- file-stixPOD

Represents the connector-import-file-stix POD.

Connector-import- file-yara POD

Represents the connector-import-file-yara POD.

Connector-mitre POD

Represents the connector-mitre POD.

Connector-opencti POD

Represents the connector-opencti POD.

Minio POD

Represents the minio POD.

Opencti POD

Represents the minio POD.

Opencti worker POD

Represents the Opencti worker POD.

Rabbitmq POD

Represents the rabbitmq POD.

Redis POD

Represents the redis POD.

Rsa-key-generator POD

Represents the rsa-key-generator POD.

Xtm-composer POD

Represents the xtm-composer POD.

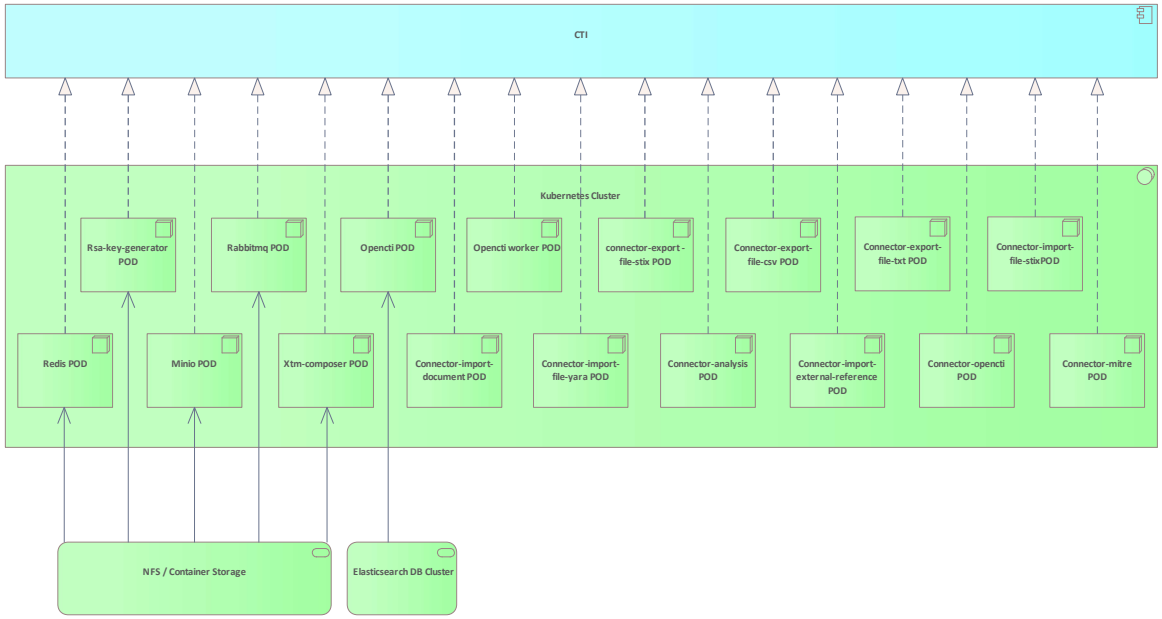


Figure 2 : Technology Usage Viewpoint CTI alt 2

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 capability to enrich detection rules, correlation mechanisms and playbooks with external threat information – including IOCs, TTPs, attacker profiling, campaigns and related vulnerabilities. In the environment of the multi-customer SOCaaS service, 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,
- Support of threat hunting and forensic analysis processes,
- Prioritization of incidents according to 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 ensures the deployment, scaling, monitoring, updating and management of containers (most often Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, the computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated management of the application life cycle.

Elasticsearch DB Cluster

Represents the technology service of the NoSQL Elasticsearch database in the form of a cluster.

Connector-analysis POD

Represents the connector-analysis POD.

Connector-export- file-csv POD

Represents the connector-export-file-csv POD.

connector-export -file-stix POD

Represents the connector-export-file-stix POD.

Connector-export- file-txt POD

Represents the connector-export-file-txt POD.

Connector-import- document POD

Represents the connector-import-document POD.

Connector-import- external-reference POD

Represents the connector-import-external-reference POD.

Connector-import- file-stixPOD

Represents the connector-import-file-stix POD.

Connector-import- file-yara POD

Represents the connector-import-file-yara POD.

Connector-mitre POD

Represents the connector-mitre POD.

Connector-opencti POD

Represents the connector-opencti POD.

Minio POD

Represents the minio POD.

Opencti POD

Represents the minio POD.

Opencti worker POD

Represents the Opencti worker POD.

Rabbitmq POD

Represents the rabbitmq POD.

Redis POD

Represents the redis POD.

Rsa-key-generator POD

Represents the rsa-key-generator POD.

Xtm-composer POD

Represents the xtm-composer POD.

1.2. Technology Viewpoint EDR/XDR

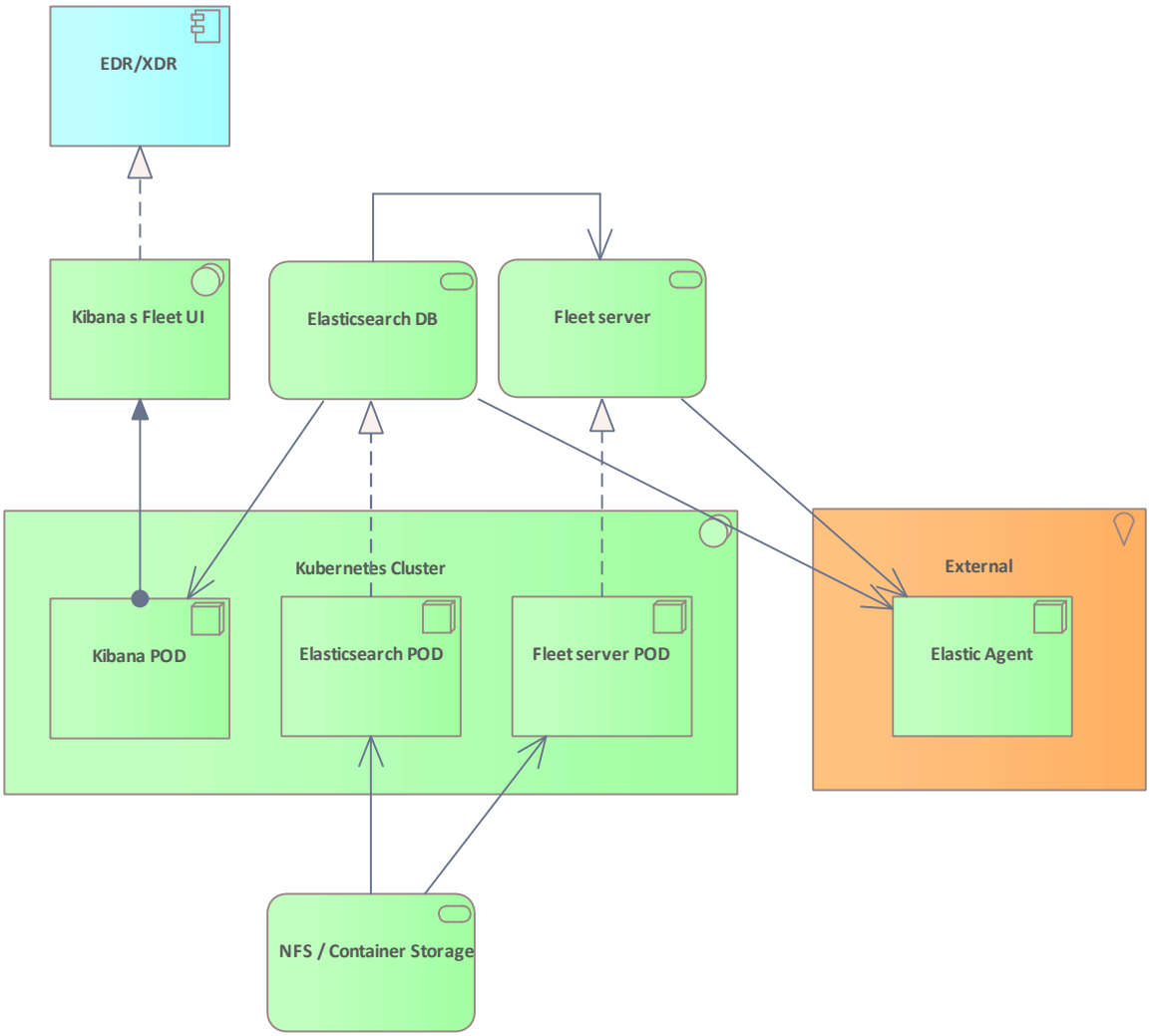


Figure 3 : Technology Usage Viewpoint EDR/XDR alt 1

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

Elasticsearch DB

Represents the technology service of the NoSQL Elasticsearch database.

Elasticsearch POD

Represents the Elasticsearch POD for the NoSQL database.

External

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

Kibana POD

Represents the Kibana POD for SIEM.

Fleet server

Represents the Fleet server technology service for managing Elastic agents.

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

Represents the Elasticsearch POD for the NoSQL database.

Kibana with Fleet UI

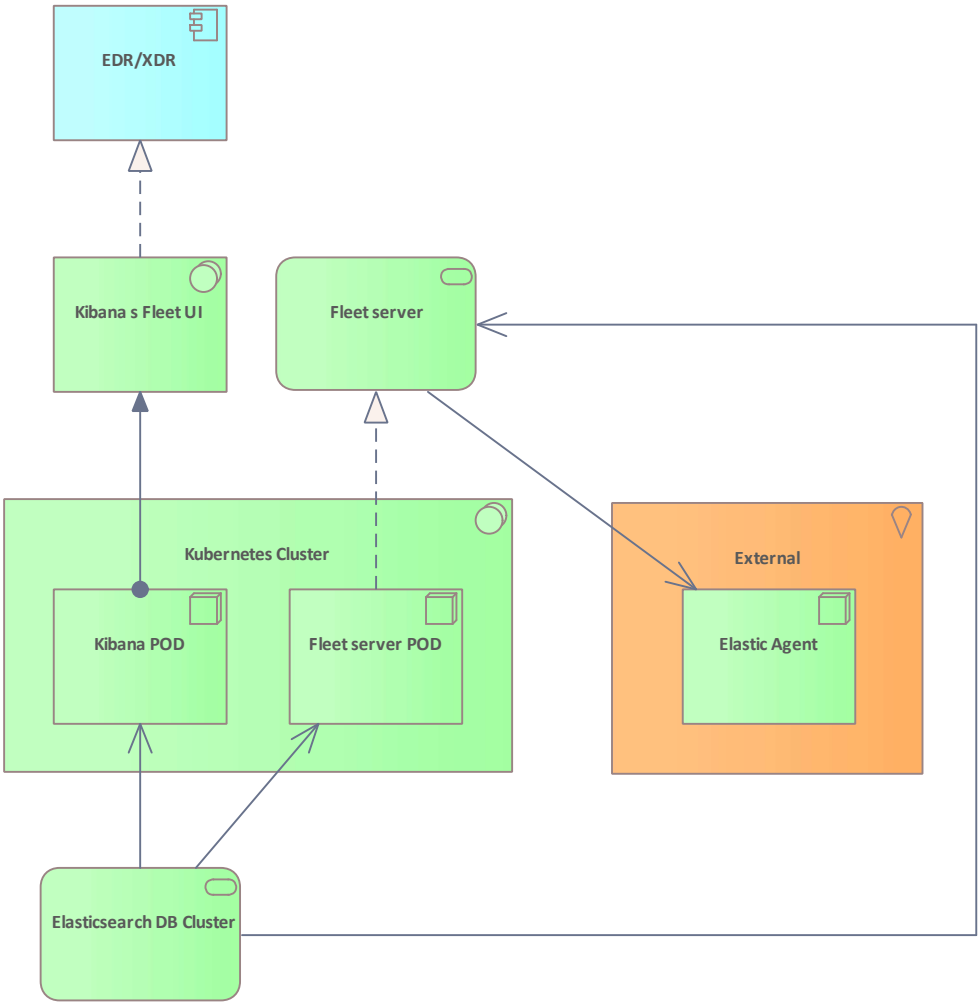


Figure 4 : Technology Usage Viewpoint EDR/XDR alt 2

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

Elasticsearch DB Cluster

Represents the technology service of the NoSQL Elasticsearch database in the form of a cluster.

External

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

Kibana POD

Represents the Kibana POD for SIEM.

Fleet server

Represents the Fleet server technology service for managing Elastic agents.

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

Represents the Elasticsearch POD for the NoSQL database.

Kibana with Fleet UI

1.3. Technology Viewpoint INNOSOC Common

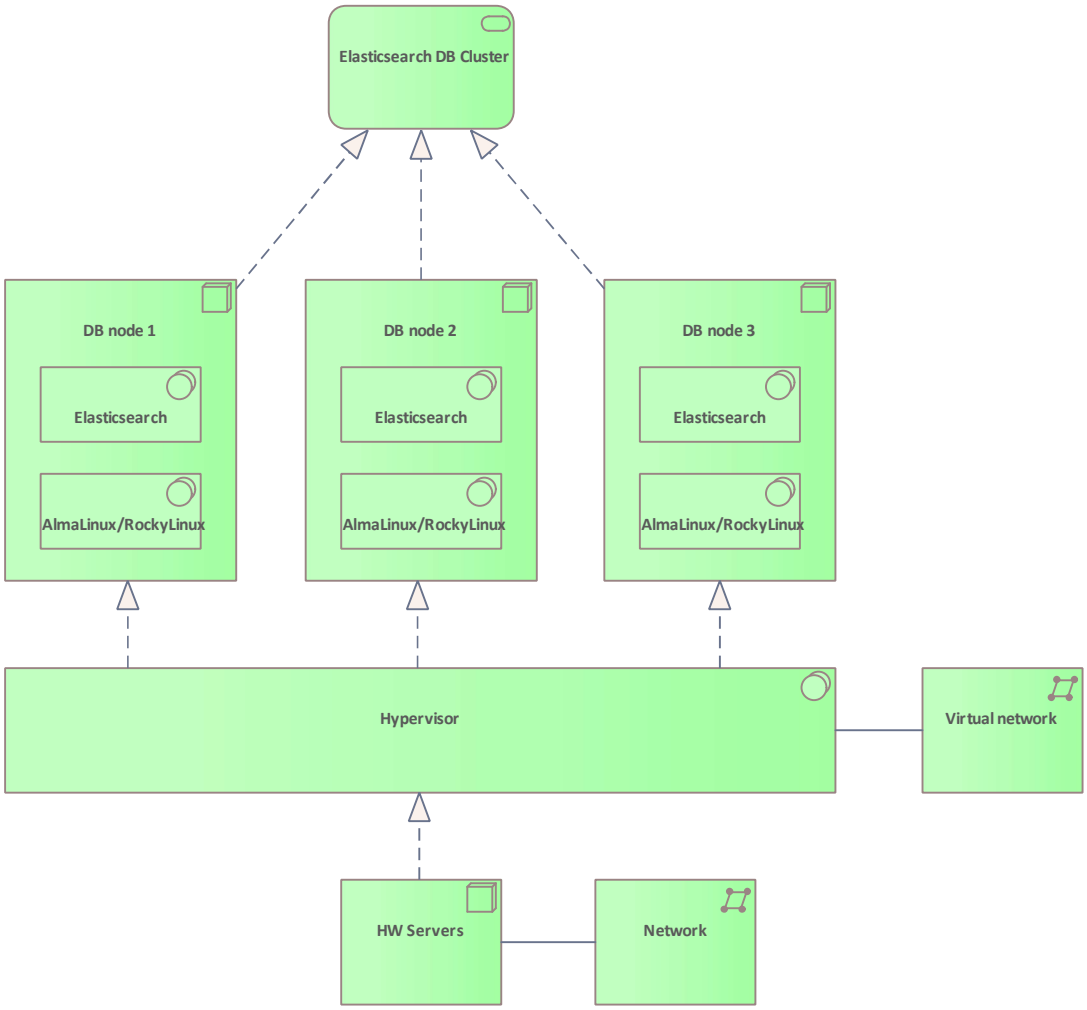


Figure 5 : Technology Viewpoint DB Cluster

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.

DB node 2

Represents Node 2 of the Elasticsearch cluster.

DB node 3

Represents Node 3 of the Elasticsearch cluster.

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

Elasticsearch

Represents the Elasticsearch NoSQL database.

AlmaLinux/RockyLinux

Represents the AlmaLinux/RockyLinux Linux distribution

Elasticsearch

Represents the Elasticsearch NoSQL database.

AlmaLinux/RockyLinux

Represents the AlmaLinux/RockyLinux Linux distribution

Elasticsearch

Represents the Elasticsearch NoSQL database.

Elasticsearch DB Cluster

Represents the technology service of the NoSQL Elasticsearch database in the form of a cluster.

1.4. Technology Viewpoint Asset Inventory

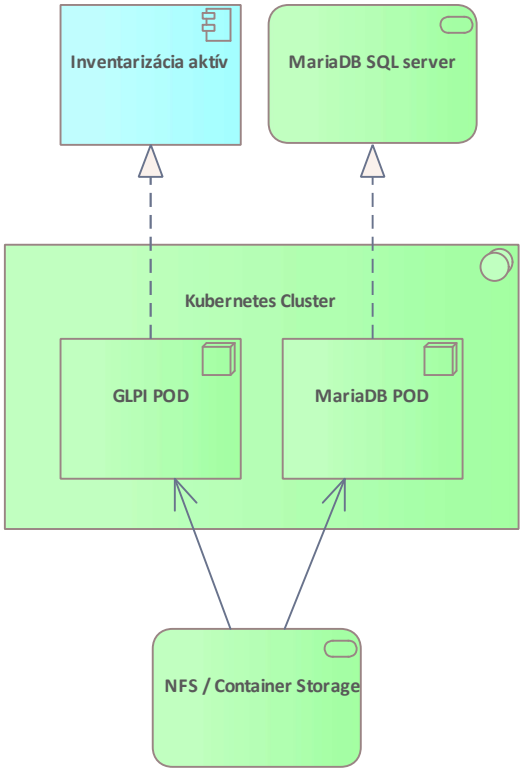


Figure 6 : Technology Usage Viewpoint Asset inventory alt 1

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

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.

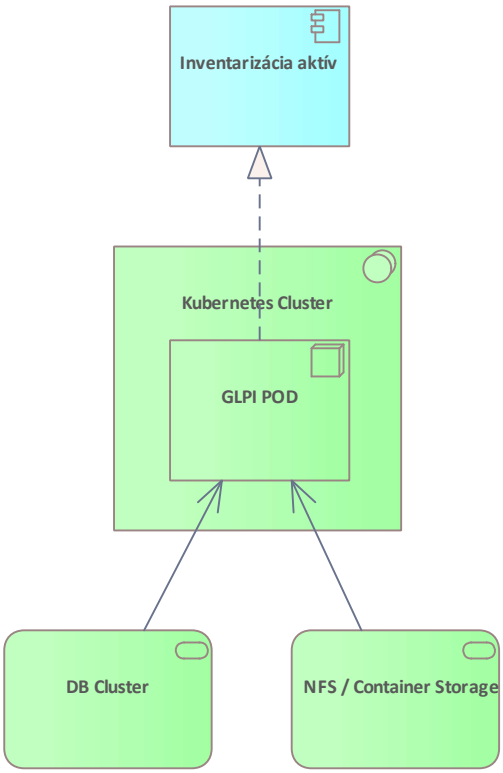


Figure 7 : Technology Usage Viewpoint Asset inventory alt 2

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

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.

1.5. Technology Viewpoint Kubernetes

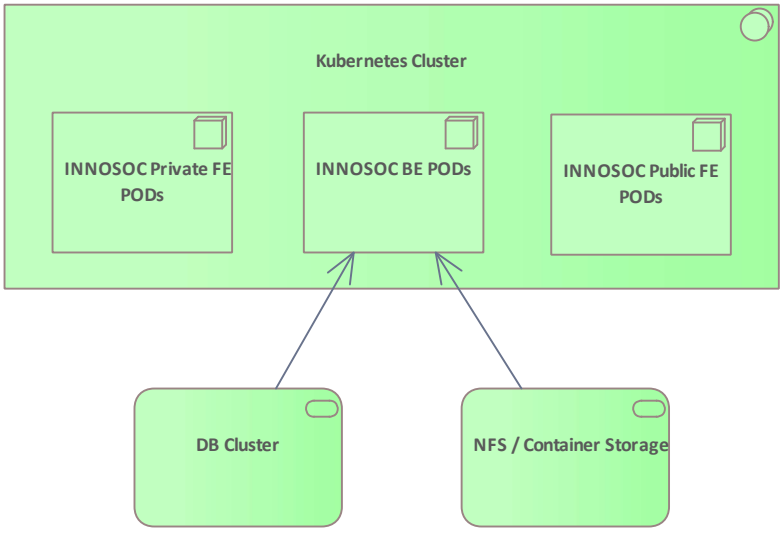


Figure 8 : Technology Usage Viewpoint CNM and Containers maping

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

INNOSOC BE PODs

Represents the CNM BE PODs. The CNM BE PODs denote container instances of the backend services (BE – Backend) of the Core Network Management (CNM) system that are deployed in a distributed environment, typically in a Kubernetes cluster or another container orchestration. These PODs form the server (logical) layer of the CNM architecture, which ensures data processing, business logic, transactions, API services and interaction with the database and network entities (core network functions).

INNOSOC Private FE PODs

The CNM FE Private PODs represent isolated and modular deployment units of the frontend components of the Customer Network Management (CNM) system 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

The CNM FE Public PODs represent computing deployment units of the frontend components of the Customer Network Management (CNM) system in a publicly accessible cloud or hybrid environment that is 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 CNM services without the need for a local installation or private infrastructure.

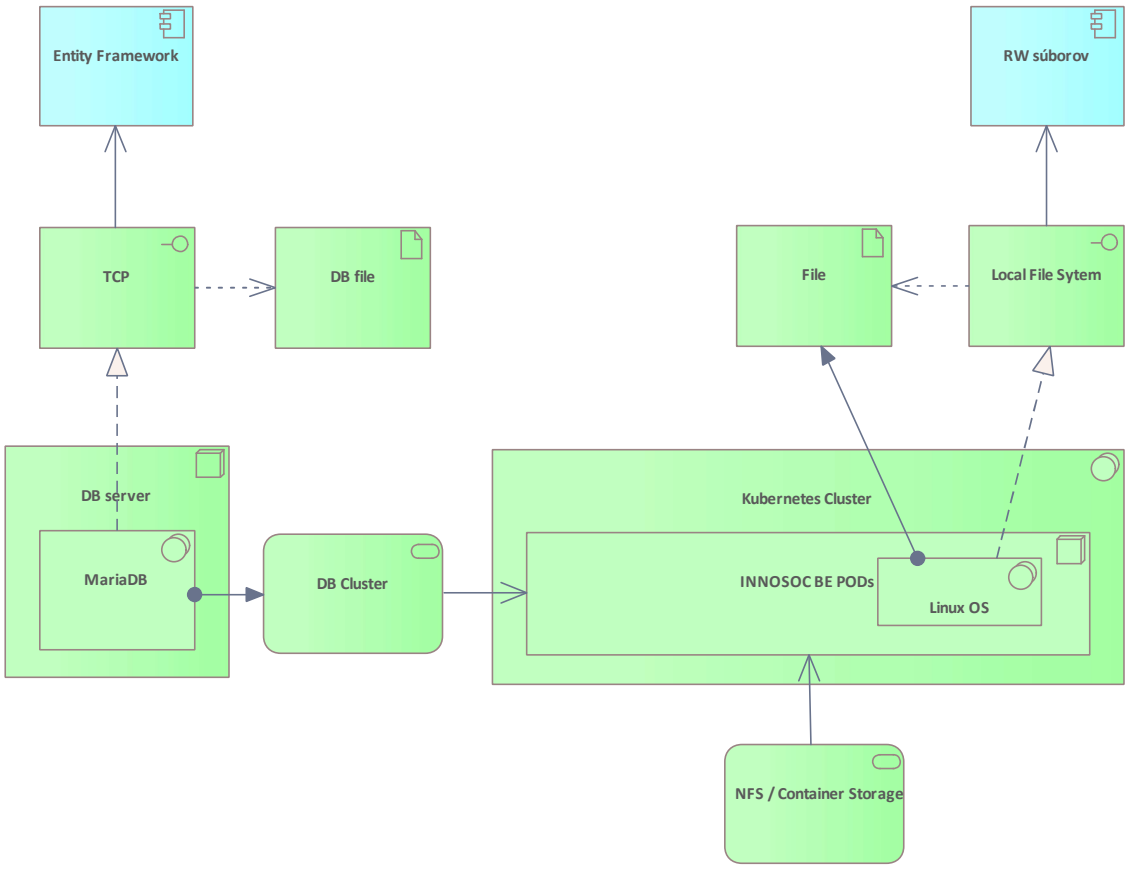


Figure 9 : Technology Usage Viewpoint CNM BE, DB and File System

The diagram represents and describes in more detail the use of the persistence layer for the needs of the BIT NC and CNM BE module by means of its Data layer component, which handles work 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. In the context of the MariaDB system, a DB file (database file) 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 often 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 the 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 ensures the deployment, scaling, monitoring, updating and management of containers (most often Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, the computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated management of the application life cycle.

INNOSOC BE PODs

Represents the CNM BE PODs. The CNM BE PODs denote container instances of the backend services (BE – Backend) of the Core Network Management (CNM) system that are deployed in a distributed environment, typically in a Kubernetes cluster or another container orchestration. These PODs form the server (logical) layer of the CNM architecture, which ensures 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.

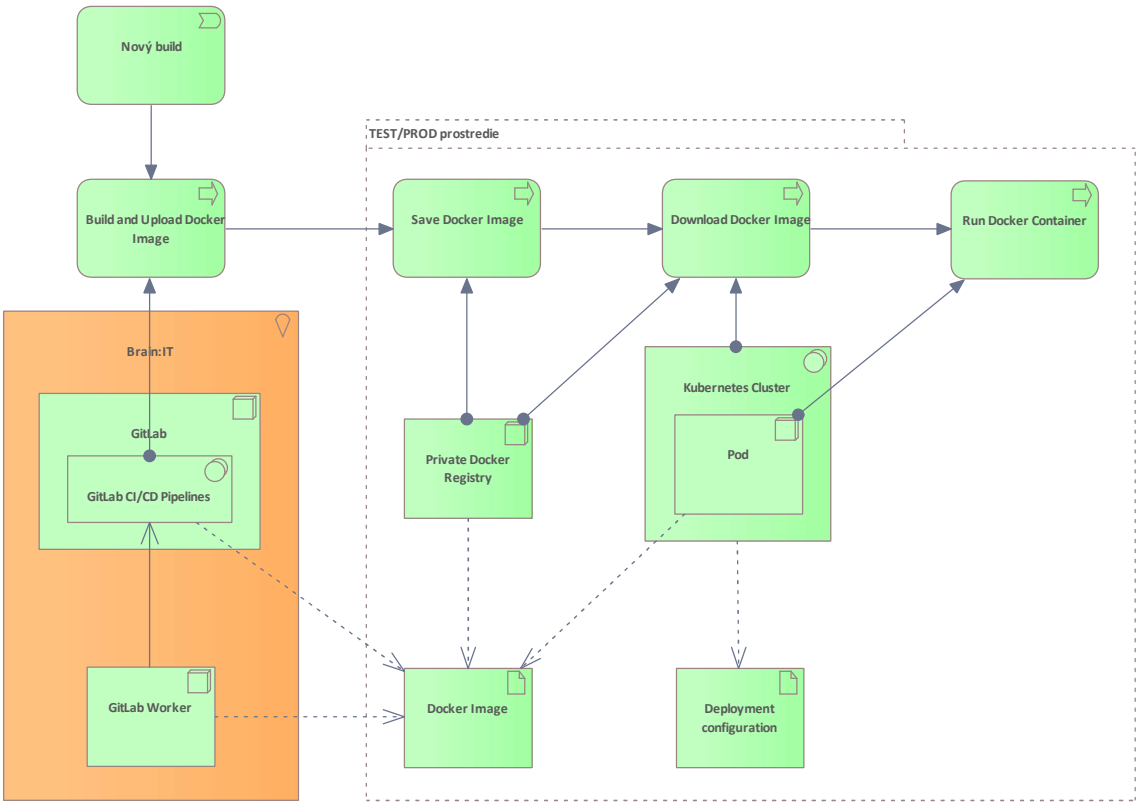


Figure 10 : Technology Viewpoint Development CI/CD TEST/PROD

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

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

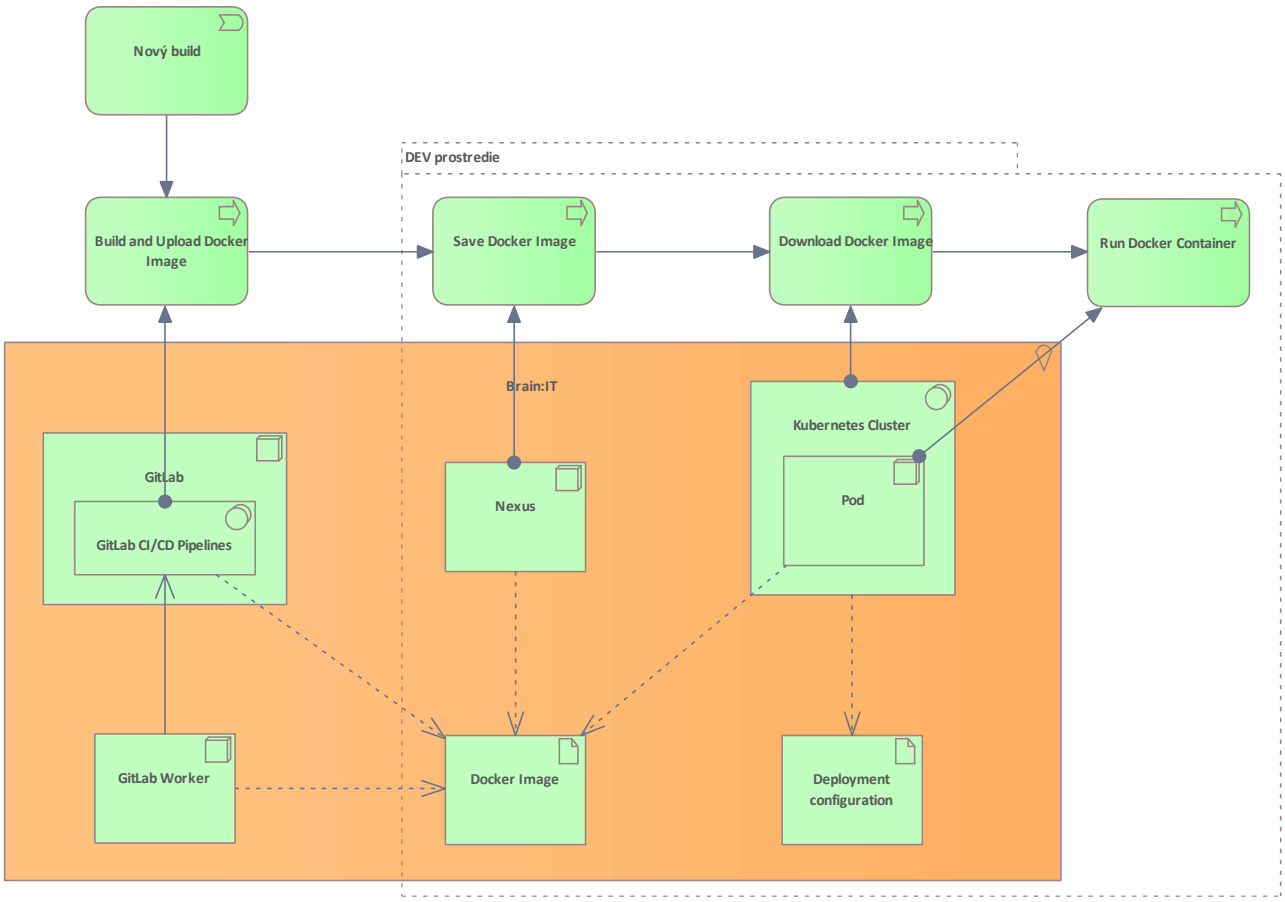


Figure 11 : Technology Viewpoint Developmnent CI/CD DEV

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

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

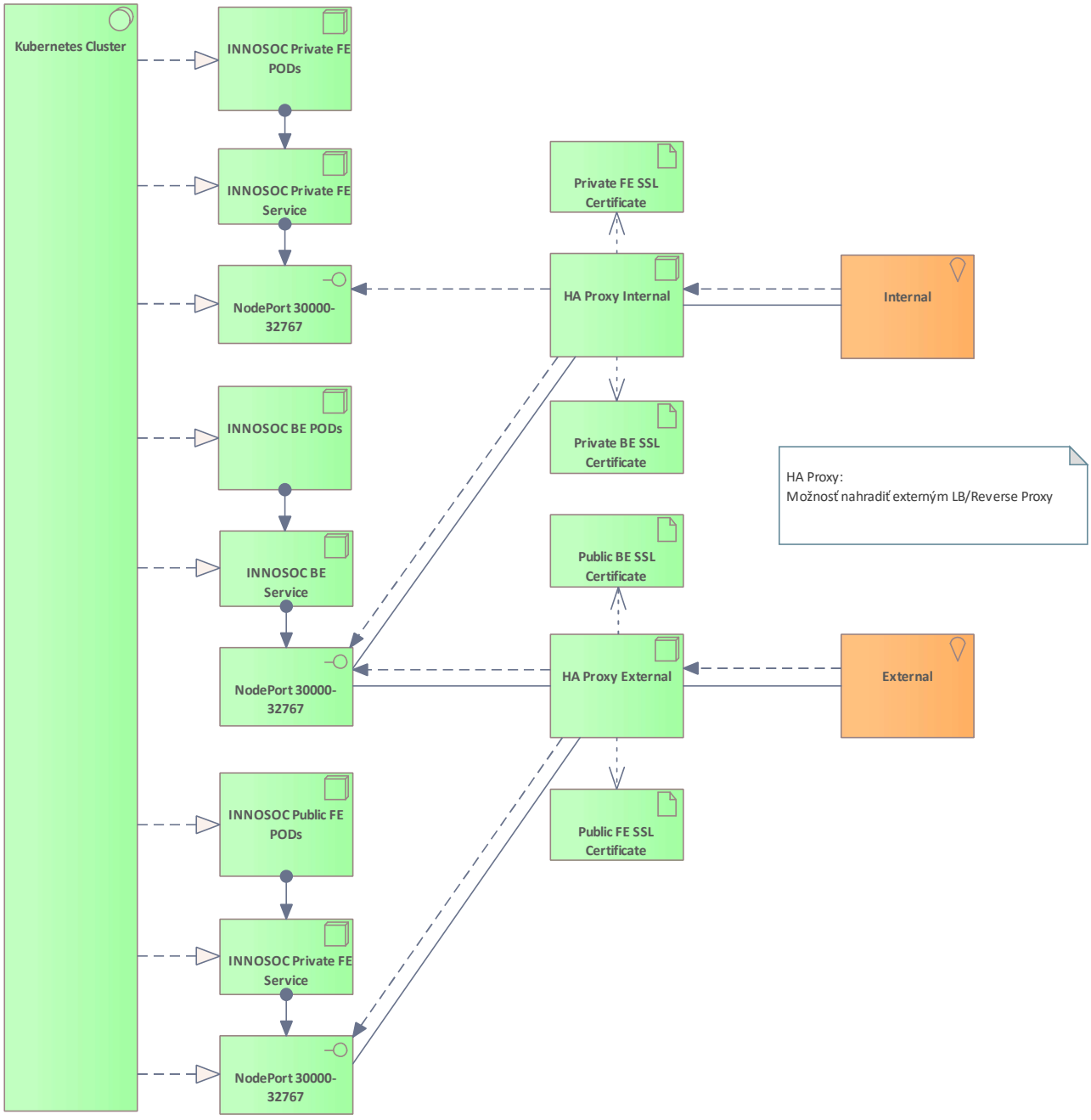


Figure 12 : Technology Viewpoint Interconnect

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:

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

INNOSOC BE PODs

Represents the CNM BE PODs. The CNM BE PODs denote container instances of the backend services (BE – Backend) of the Core Network Management (CNM) system that are deployed in a distributed environment, typically in a Kubernetes cluster or another container orchestration. These PODs form the server (logical) layer of the CNM architecture, which ensures data processing, business logic, transactions, API services and interaction with the database and network entities (core network functions).

INNOSOC Private FE PODs

The CNM FE Private PODs represent isolated and modular deployment units of the frontend components of the Customer Network Management (CNM) system 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

The CNM FE Public PODs represent computing deployment units of the frontend components of the Customer Network Management (CNM) system in a publicly accessible cloud or hybrid environment that is 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 CNM services without the need for a local installation or private infrastructure.

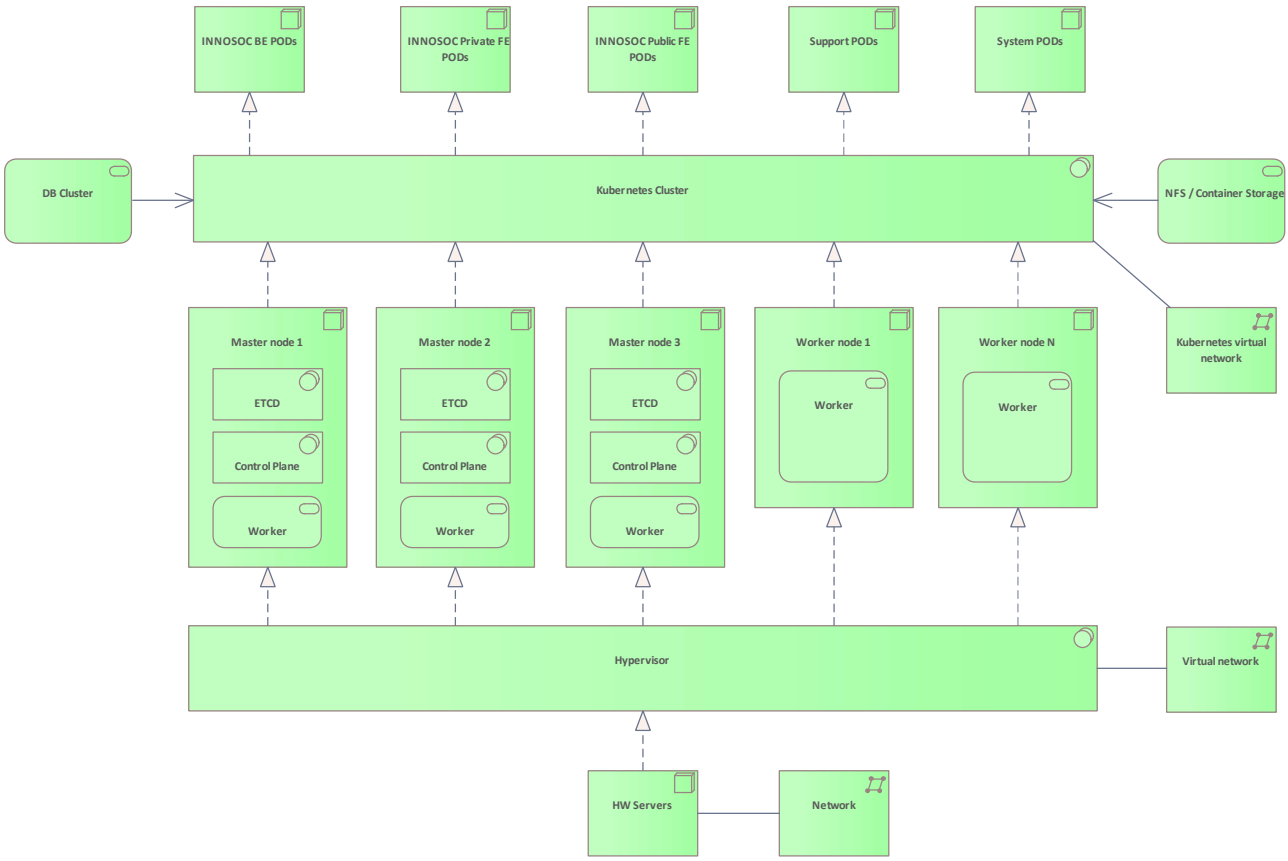


Figure 13 : Technology Viewpoint Kubernetes Cluster

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

Master node 3

Represents the Master node of the Kubernetes cluster

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

Kubernetes Cluster

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

INNOSOC BE PODs

Represents the CNM BE PODs. The CNM BE PODs denote container instances of the backend services (BE – Backend) of the Core Network Management (CNM) system that are deployed in a distributed environment, typically in a Kubernetes cluster or another container orchestration. These PODs form the server (logical) layer of the CNM architecture, which ensures data processing, business logic, transactions, API services and interaction with the database and network entities (core network functions).

INNOSOC Private FE PODs

The CNM FE Private PODs represent isolated and modular deployment units of the frontend components of the Customer Network Management (CNM) system 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

The CNM FE Public PODs represent computing deployment units of the frontend components of the Customer Network Management (CNM) system in a publicly accessible cloud or hybrid environment that is 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 CNM 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

Control Plane

Represents the Control Plane for managing the Kubernetes cluster

ETCD

Represents the ETCD database

Worker

Control Plane

Represents the Control Plane for managing the Kubernetes cluster

ETCD

Represents the ETCD database

Worker

Worker

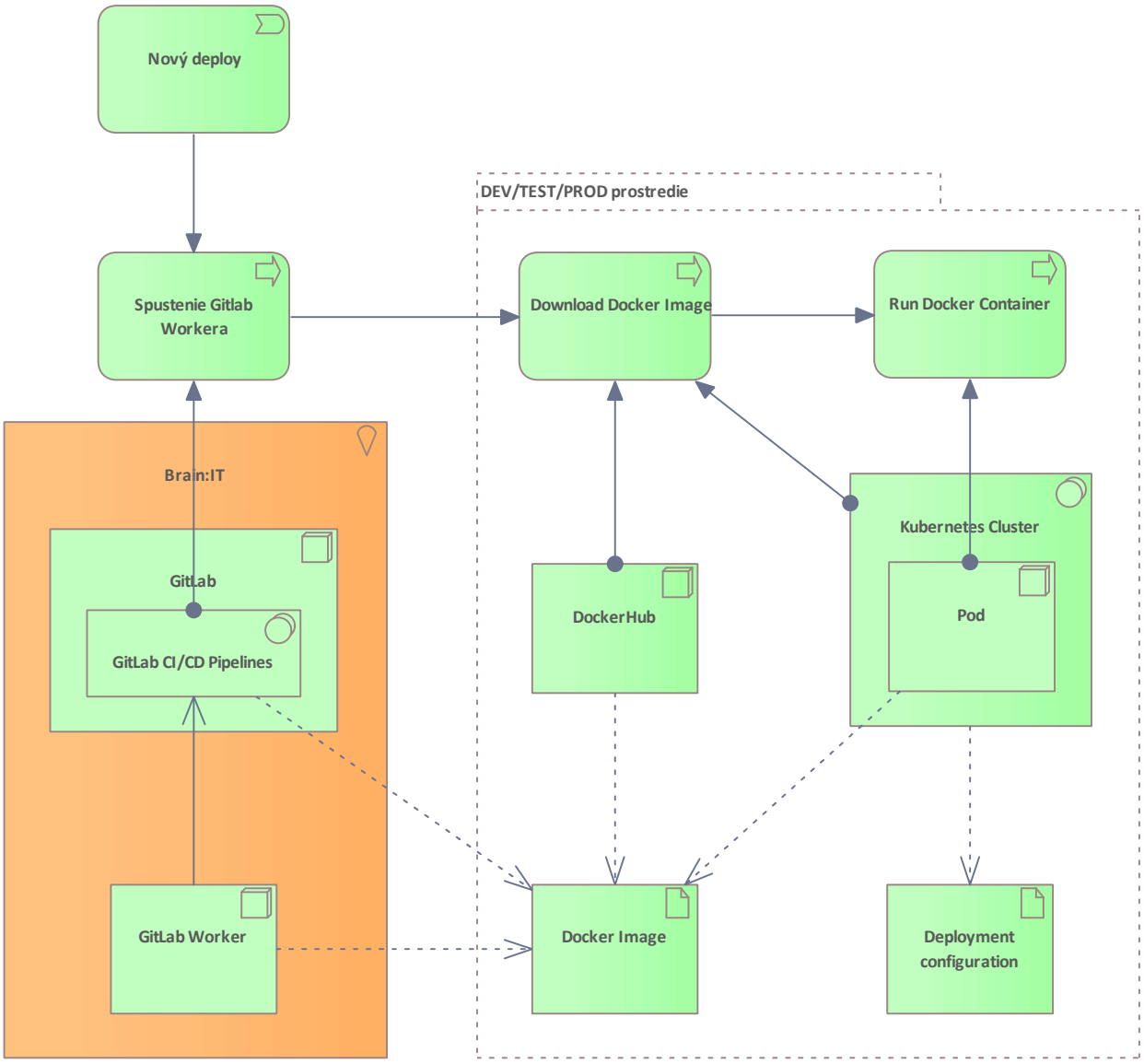


Figure 14 : Technology Viewpoint standard CI/CD DEV/TEST/PROD

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

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

1.6. Technology Viewpoint OpenStack

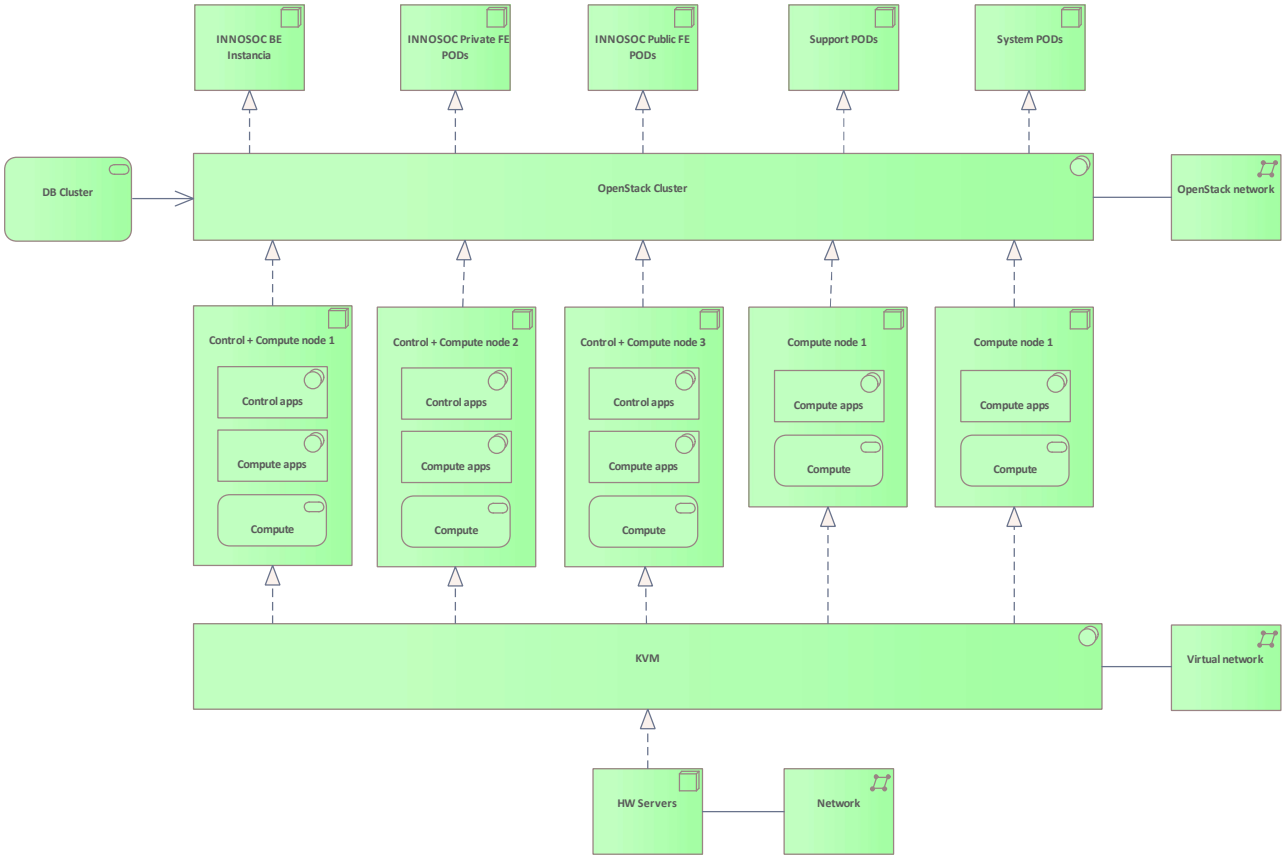


Figure 15 : Technology Viewpoint OpenStack Cluster

Description of the elements used:

DB Cluster

Represents the clustered relational database service

Compute node 1

Represents a Compute node of the OpenStack cluster

Compute node 1

Represents a Compute node of the OpenStack cluster

Control + Compute node 1

Represents Control and Compute node 1 of the OpenStack cluster

Control + Compute node 2

Represents Control and Compute node 2 of the OpenStack cluster

Control + Compute node 3

Represents Control and Compute node 3 of the OpenStack cluster

HW Servers

Represents the HW server infrastructure

KVM

Represents a type 1 KVM hypervisor.

Network

Represents the network/subnet of the HW infrastructure

OpenStack Cluster

Represents the OpenStack Cluster

OpenStack network

Represents the network/subnet of the OpenStack cluster, specifically:

Corporate network

OpenStack Tenant Data Network

OpenStack Internal Admin / API Subnet

Support PODs

Represents the PODs of supporting services

System PODs

Represents the PODs of system services

Virtual network

Represents the virtual network/subnet of the hypervisor

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 CNM FE Private PODs

INNOSOC Public FE PODs

Represents the CNM FE Public PODs

Compute

Represents the role and service of the Compute node

Compute apps

Represents the set of Compute applications

Control apps

Represents the set of applications for the Control node required for managing the cluster

Compute

Represents the role and service of the Compute node

Compute apps

Represents the set of Compute applications

Compute

Represents the role and service of the Compute node

Compute apps

Represents the set of Compute applications

Control apps

Represents the set of applications for the Control node required for managing the cluster

Compute

Represents the role and service of the Compute node

Compute apps

Represents the set of Compute applications

Control apps

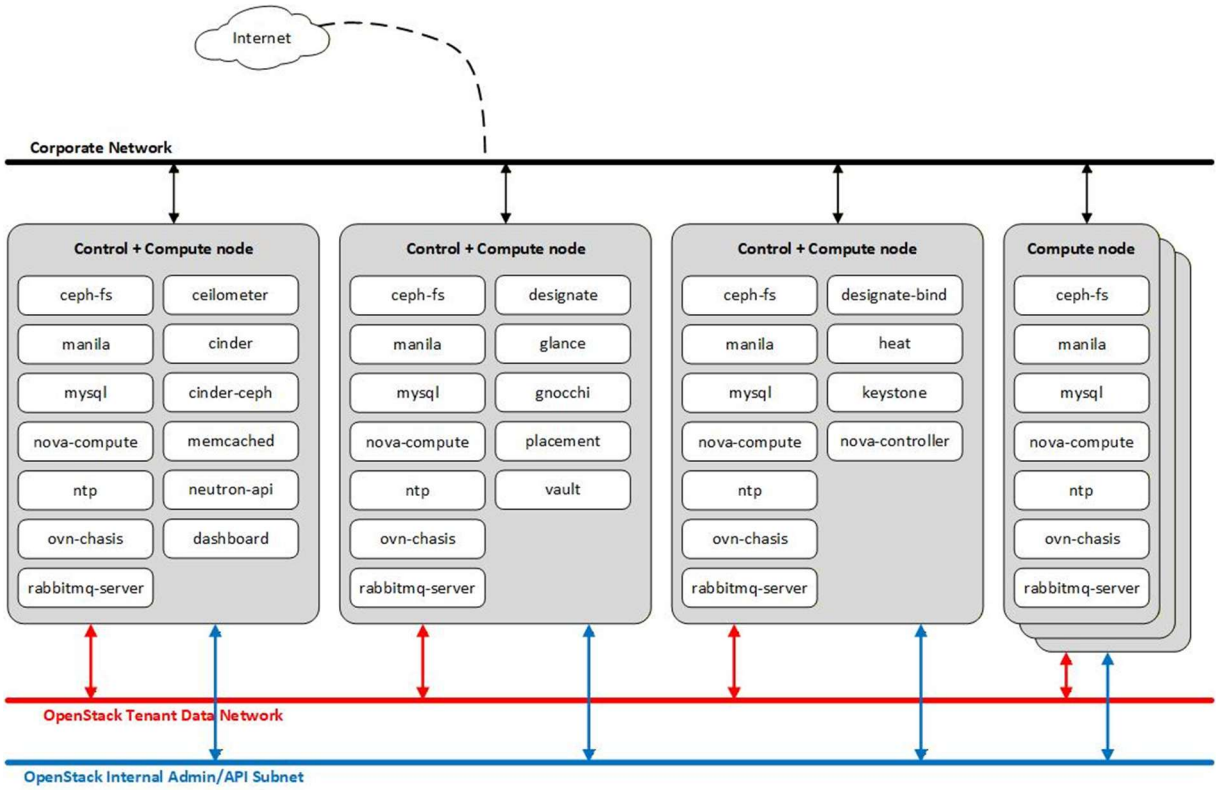
Represents the set of applications for the Control node required for managing the cluster

Compute

Represents the role and service of the Compute node

Compute apps

Represents the set of Compute applications



1.7. Technology Viewpoint Reporting

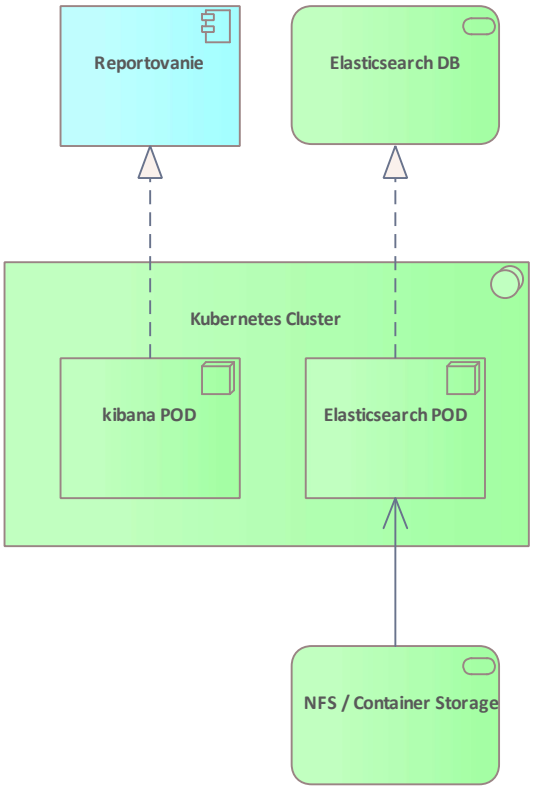


Figure 16 : Technology Usage Viewpoint Reporting alt 1

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 for incidents, alerts, SLAs and KPIs,

• Overviews of trends over time and assessment 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 ensures the deployment, scaling, monitoring, updating and management of containers (most often Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, the computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated management of the application life cycle.

Elasticsearch DB

Represents the technology service of the NoSQL Elasticsearch database.

Elasticsearch POD

Represents the Elasticsearch POD for the NoSQL database.

kibana POD

Represents the Kibana POD for Reporting.

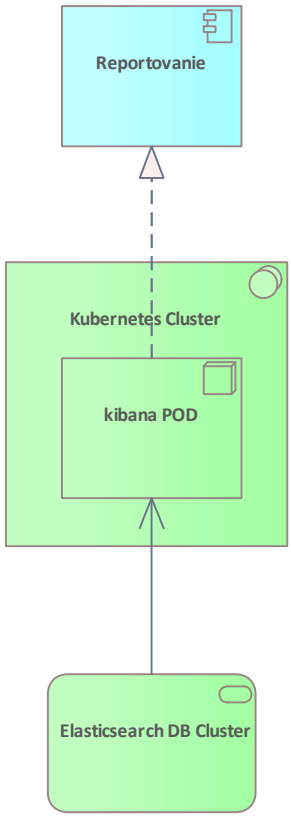


Figure 17 : Technology Usage Viewpoint Reporting alt 2

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 for incidents, alerts, SLAs and KPIs,

• Overviews of trends over time and assessment 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 ensures the deployment, scaling, monitoring, updating and management of containers (most often Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, the computing resources (CPU, memory, storage, network) are abstracted and managed as a single system, which enables automated management of the application life cycle.

Elasticsearch DB Cluster

Represents the technology service of the NoSQL Elasticsearch database in the form of a cluster.

kibana POD

Represents the Kibana POD for Reporting.

1.8. Technology Viewpoint SIEM

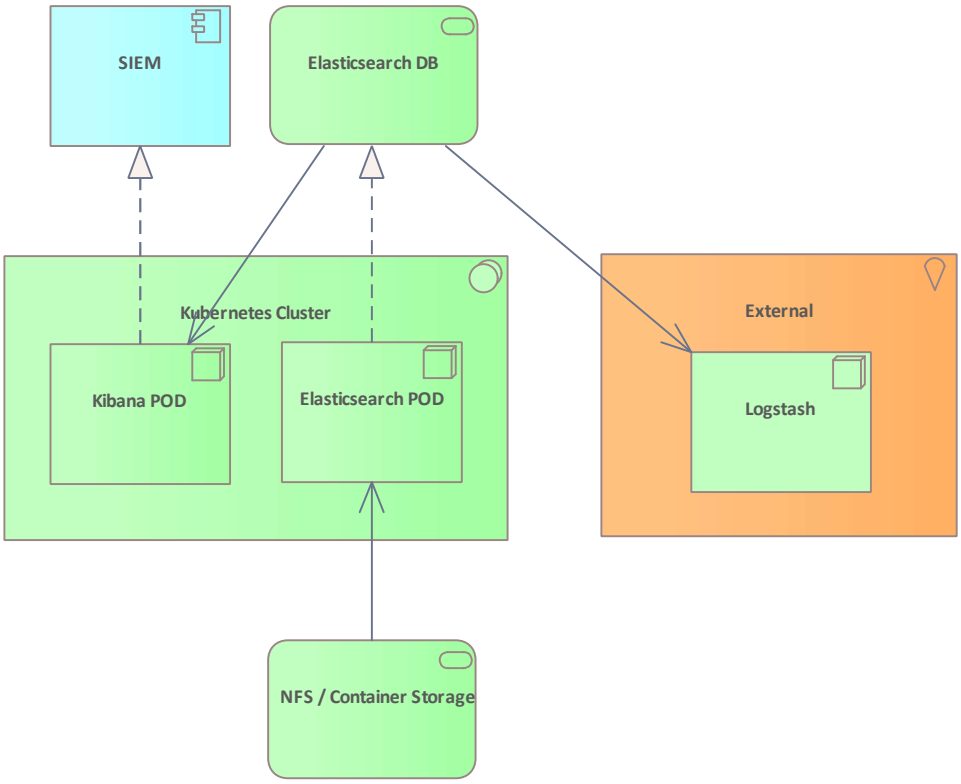


Figure 18 : Technology Usage Viewpoint SIEM alt 1

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

Elasticsearch DB

Represents the technology service of the NoSQL Elasticsearch database.

Elasticsearch POD

Represents the Elasticsearch POD for the NoSQL database.

External

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

Kibana POD

Represents the Kibana POD for SIEM.

Logstash

Represents a Logstashg instance for processing logs and writing them to Elasticsearch.

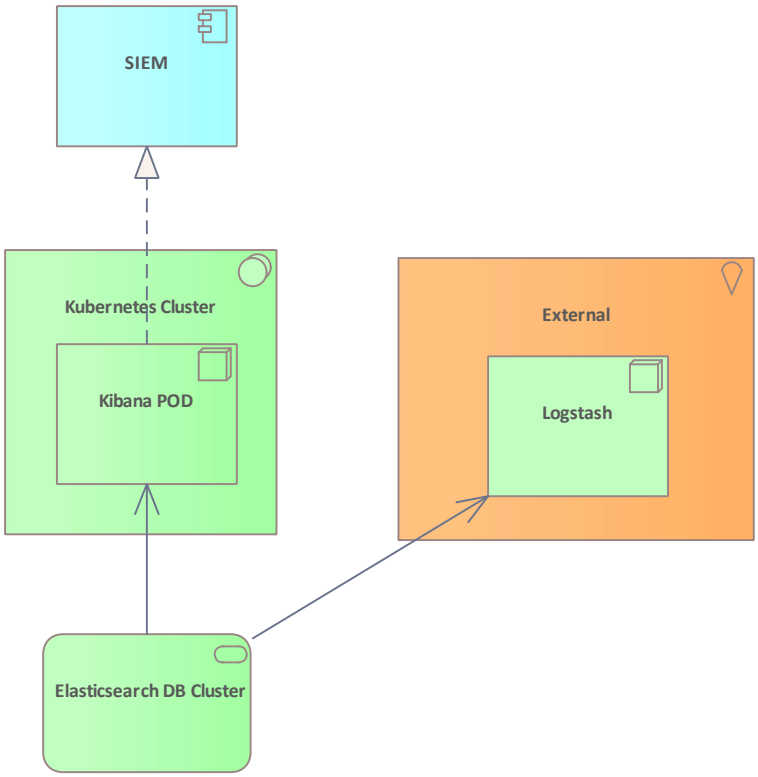


Figure 19 : Technology Usage Viewpoint SIEM alt 2

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

Elasticsearch DB Cluster

Represents the technology service of the NoSQL Elasticsearch database in the form of a cluster.

External

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

Kibana POD

Represents the Kibana POD for SIEM.

Logstash

Represents a Logstashg instance for processing logs and writing them to Elasticsearch.

1.9. Technology Viewpoint Vulnerability Scanner

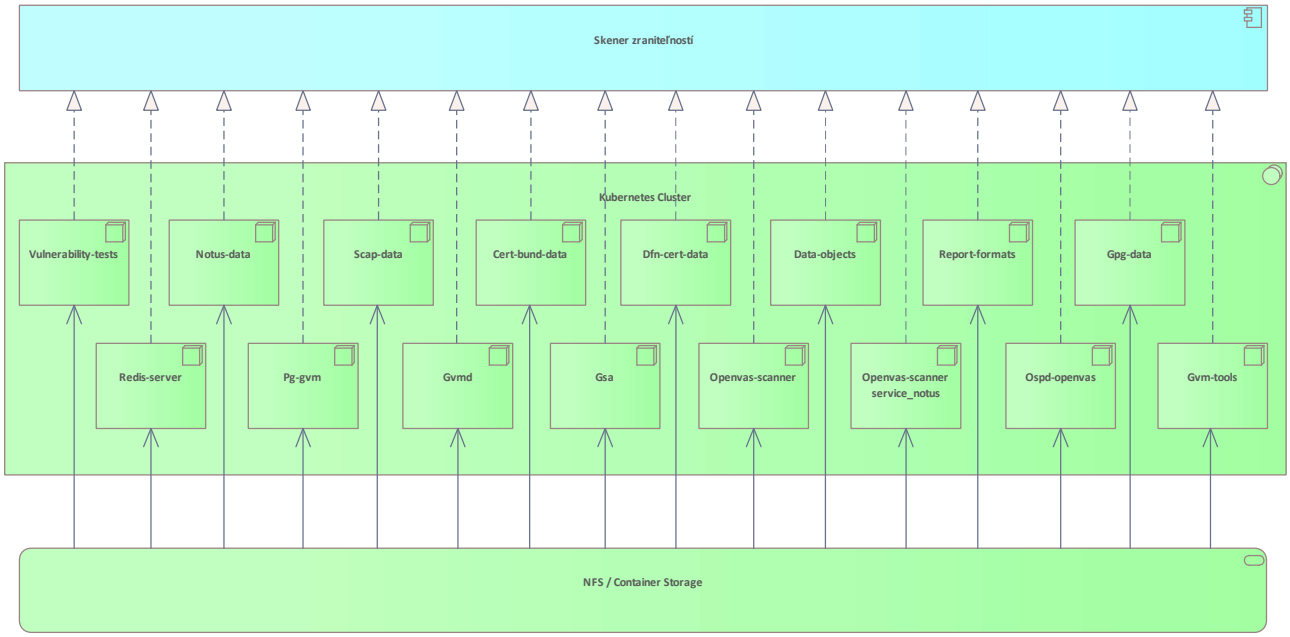


Figure 20 : Technology Usage Viewpoint Vulnerability scanner

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

Cert-bund-data

Represents the cert-bund-data POD.

Data-objects

Represents the data-objects POD.

Dfn-cert-data

Represents the dfn-cert-data POD.

Gpg-data

Represents the gpg-data POD.

Gsa

Represents the gsa POD.

Gvmd

Represents the gvmd POD.

Gvm-tools

Represents the gvm-tools POD.

Notus-data

Represents the notus-data POD.

Openvas-scanner

Represents the openvas-scanner POD.

Openvas-scanner service_notus

Represents the openvas-scanner service_notus POD.

Ospd-openvas

Represents the ospd-openvas POD.

Pg-gvm

Represents the pg-gvm POD.

Redis-server

Represents the redis-server POD.

Report-formats

Represents the report-formats POD.

Scap-data

Represents the scap-data POD.

Vulnerability-tests

Represents the vulnerability-tests POD.

1.10. Technology Viewpoint SOAR

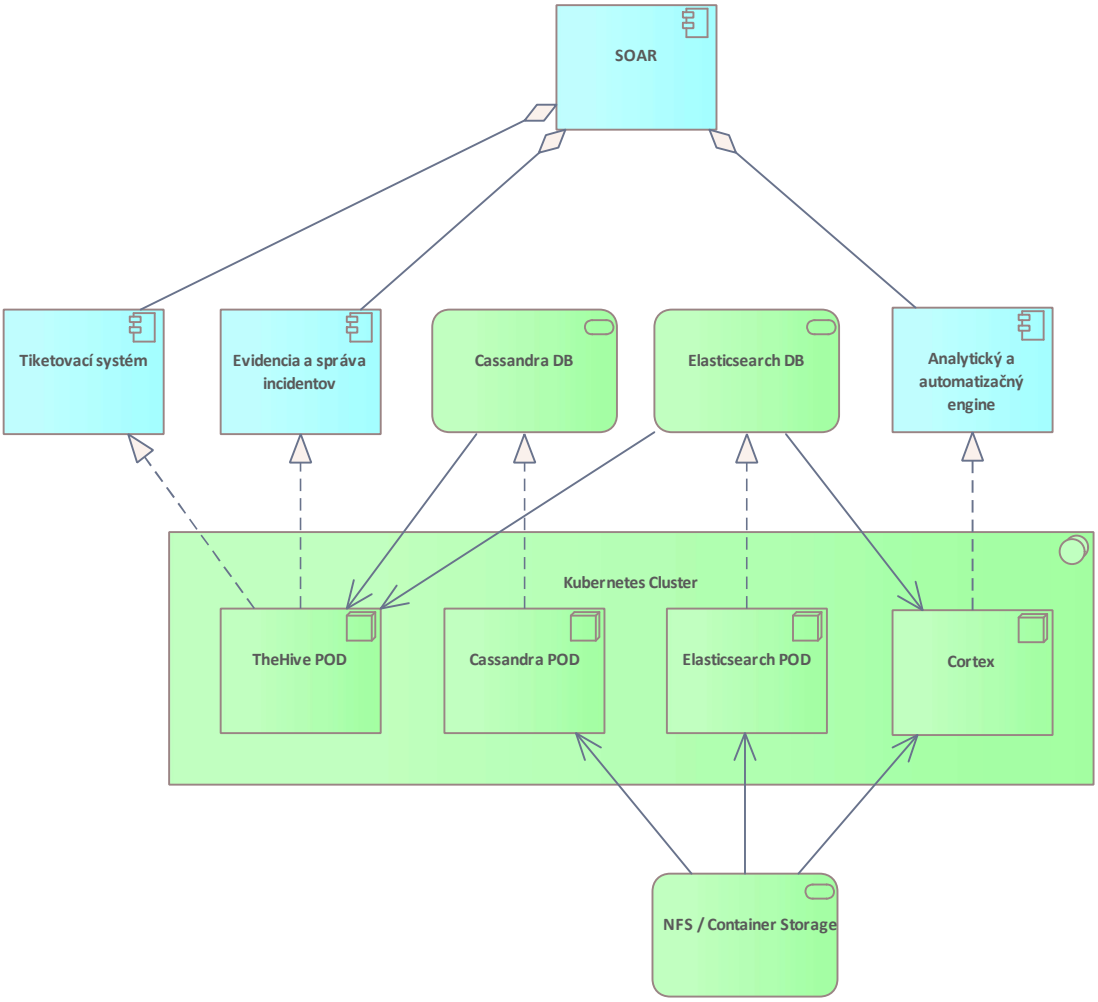


Figure 21 : Technology Usage Viewpoint SOAR alt 1

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 the NoSQL databases for Cassandra and Elasticsearch.

Description of the elements used:

Analytics and automation engine

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 ensures the deployment, scaling, monitoring, updating and management of containers (most often Docker or compatible OCI containers) across a group of physical or virtual servers. In a Kubernetes cluster, the computing resources (CPU, memory, storage, network) are

abstracted and managed as a single system, which enables automated management of the application life cycle.

Cassandra DB

Represents the technology service of the NoSQL Cassandra database.

Elasticsearch DB

Represents the technology service of the NoSQL Elasticsearch database.

Cassandra POD

Represents the Apache Cassandra POD for the NoSQL database.

Elasticsearch POD

Represents the Elasticsearch POD for the NoSQL database.

TheHive POD

Represents the TheHive POD for asset inventory.

Cortex

Represents the Cortex POD for SOAR.

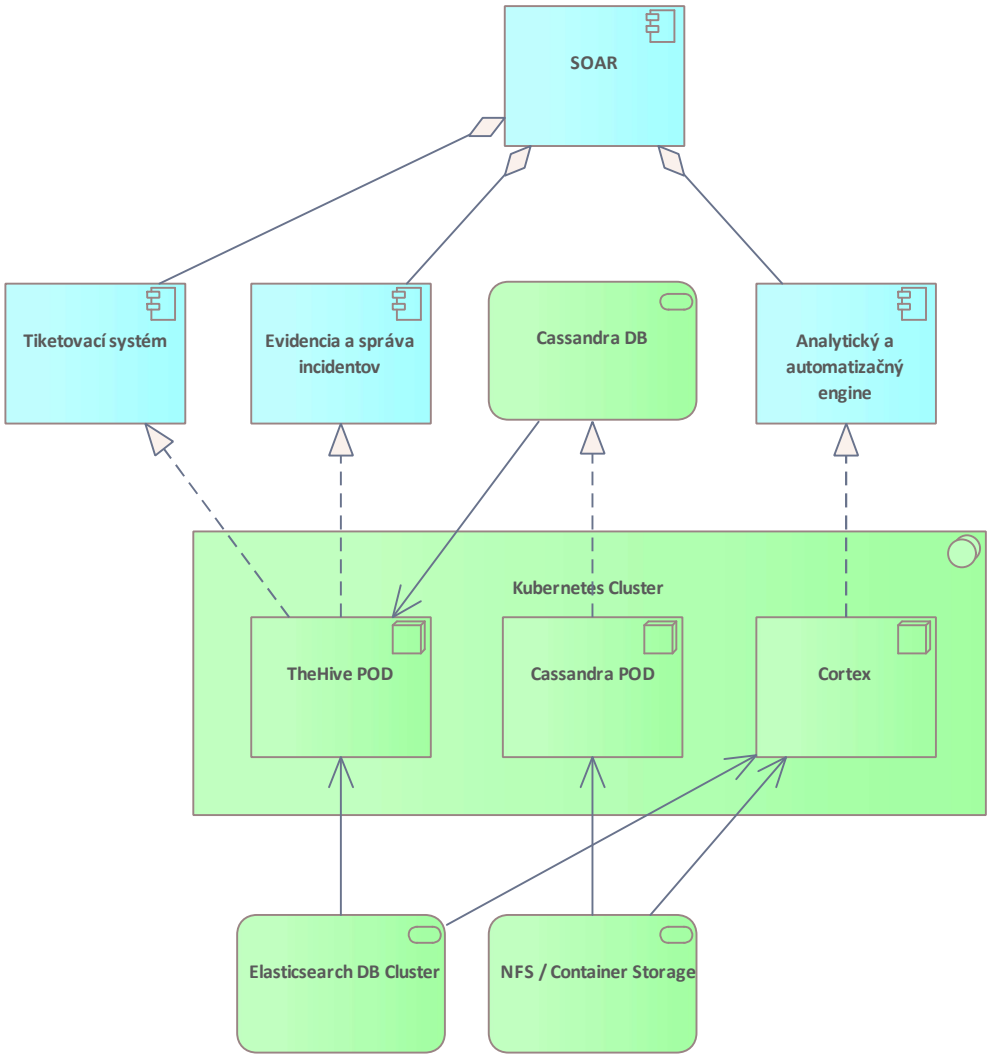


Figure 22 : Technology Usage Viewpoint SOAR alt 2

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

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

Cassandra DB

Represents the technology service of the NoSQL Cassandra database.

Elasticsearch DB Cluster

Represents the technology service of the NoSQL Elasticsearch database in the form of a cluster.

Cassandra POD

Represents the Apache Cassandra POD for the NoSQL database.

TheHive POD

Represents the TheHive POD for asset inventory.

Cortex

Represents the Cortex POD for SOAR.

1.11. Technology Viewpoint Ticketing system / Incident registration and management

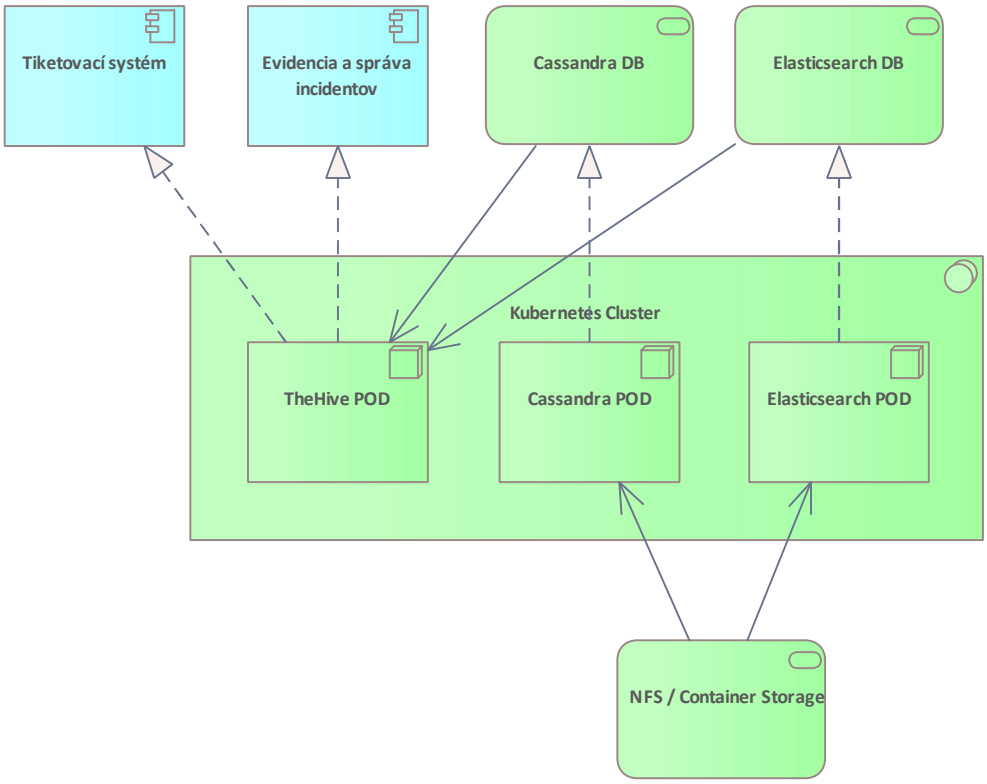


Figure 23 : Technology Usage Viewpoint Ticketing system / Incident registration and management alt 1

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

Cassandra DB

Represents the technology service of the NoSQL Cassandra database.

Elasticsearch DB

Represents the technology service of the NoSQL Elasticsearch database.

Cassandra POD

Represents the Apache Cassandra POD for the NoSQL database.

Elasticsearch POD

Represents the Elasticsearch POD for the NoSQL database.

TheHive POD

Represents the TheHive POD for asset inventory.

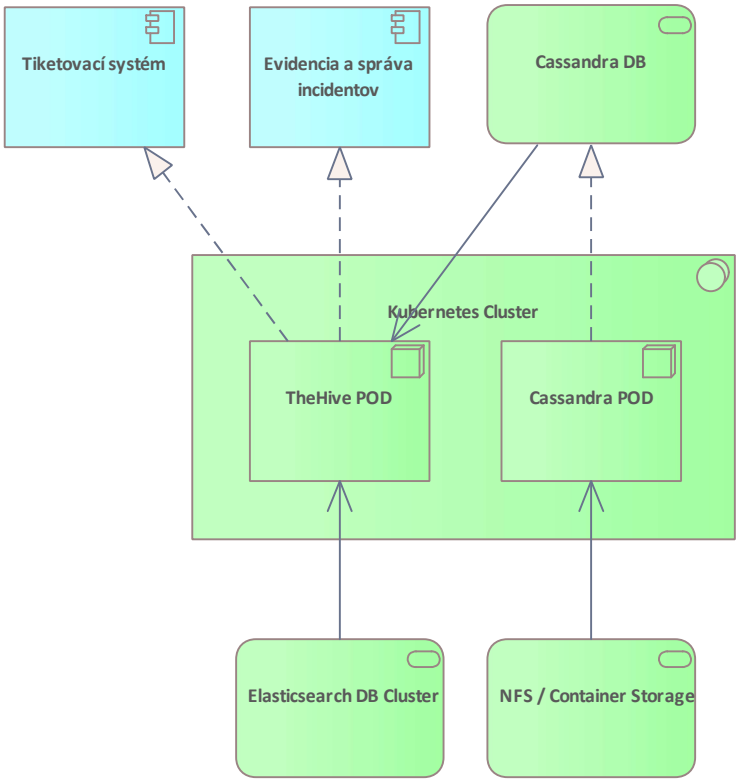


Figure 24 : Technology Usage Viewpoint Ticketing system / Incident registration and management alt 2

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

Cassandra DB

Represents the technology service of the NoSQL Cassandra database.

Elasticsearch DB Cluster

Represents the technology service of the NoSQL Elasticsearch database in the form of a cluster.

Cassandra POD

Represents the Apache Cassandra POD for the NoSQL database.

TheHive POD

Represents the TheHive POD for asset inventory.

1.12. Technology Viewpoint Knowledge Portal for the SOC Team

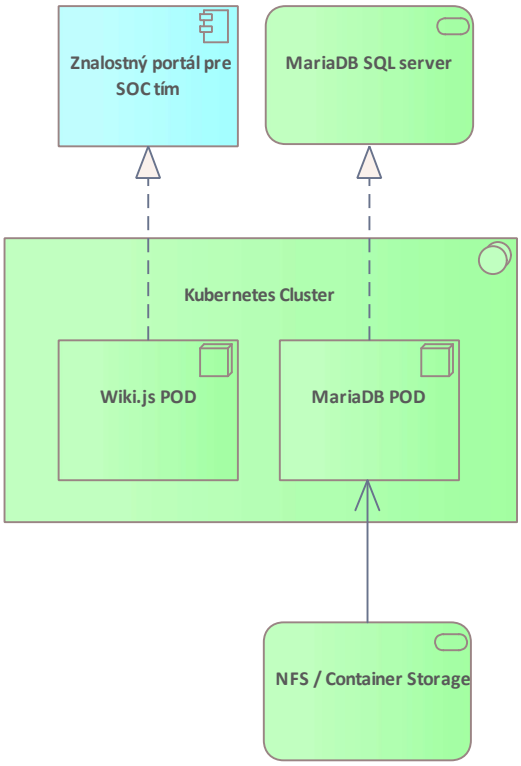


Figure 25 : Technology Usage Viewpoint Knowledge portal for the SOC team alt 1

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

MariaDB SQL server

Represents the technology service of the relational MariaDB SLQ database.

MariaDB POD

Represents the MariaDB POD for the relational SQL DB.

Wiki.js POD

Represents the Wiki.js POD for the Knowledge portal for the SOC team.

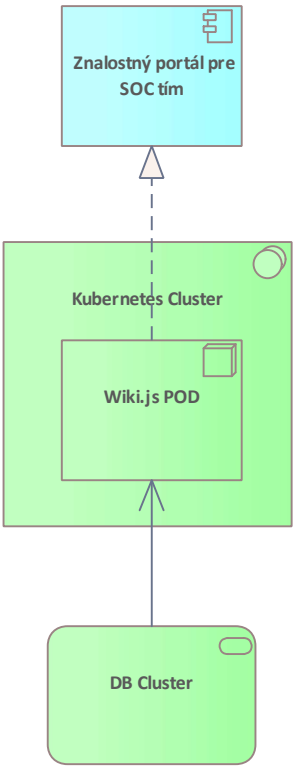


Figure 26 : Technology Usage Viewpoint Knowledge portal for the SOC team alt 2

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

Wiki.js POD

Represents the Wiki.js POD for the Knowledge portal for the SOC team.

3. CONCLUSION

The solution designed within the INNOSOC project represents a comprehensive and innovative approach to building a multi-tenant cybersecurity operations centre provided as a cloud service. The work described in detail the technological components, architectural layers and supporting infrastructure which together form the basis for a highly scalable, secure and flexible environment. The deployment of automation elements in combination with the integration of machine learning methods, in particular deep neural networks, significantly increases the system's ability to respond to current threats and to identify anomalies effectively in real time.

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

The chosen technological approach reflects current requirements for modern security architectures and at the same time enables further development of the system with an emphasis on modularity, interoperability and operational efficiency. The result is a robust solution capable of responding flexibly to changing conditions in the field of cybersecurity, while maintaining a high level of automation, observability and management of multiple client domains within a single centralized environment.

The architecture created respects the principles of the modern DevSecOps approach, which guarantees the integration of security mechanisms directly into the life cycle of service development and operation. The emphasis on containerization, the use of orchestration via Kubernetes and the management of CI/CD processes guarantees the repeatability, reliability and consistency of service deployment in environments with high demands on availability and security.

From the infrastructure perspective, the solution was designed to support a high degree of isolation, multi-tenant management and easy extensibility through scalable deployments and private or public integration layers. Technological elements such as private and public Docker Registries, valid SSL certificates, HAProxy reverse proxy layers and centralized control via GitLab CI/CD provide a stable and secure foundation for the operation of sensitive security services.

In conclusion, it can be stated that the proposed system architecture meets the requirements for a modern cybersecurity operations centre, combining high technical maturity with an emphasis on efficiency, reliability and sustainability in a dynamic cyber environment. The system is ready for further extension with advanced analytical modules, direct connection to external security sources and the development of autonomous response mechanisms, thereby creating a strong foundation for future innovative security solutions.