EXTRACT

A distributed data-mining software platform for extreme data across the compute continuum

D4.2 First Release of the Compute Continuum and Final Integration Plan

Version 1.0

Documentation Information

Contract Number	**101093110**
Project Website	www.extract-project.eu
Contractual Deadline	M15, 31st of March 2024
Dissemination Level	PU
Nature	O
Author	BSC
Contributors	BIN, IBM, IKL, SIX, URV
Reviewer	SIX
Keywords	Compute Continuum, Requirements, Integration Plan, HPC

The EXTRACT Project has received funding from the European Union's Horizon Europe programme under grant agreement number 101093110.

Change Log

Version	Description Change
V0.1	BSC: Initial Draft and structure
V0.2	First BSC's Contributions and Structure adapted
V0.3	Reordering + IBM content
V0.4	Sky Store Demonstrator (IBM)
V0.9	Review Acceptance from SIX
V1.0	Final review BSC. Submitted Version

Table of Contents

1. Executive Summary

This deliverable presents the first release of the Compute Continuum for EXTRACT, a significant milestone in this WP's goal to realize a seamless integration of computing resources across Edge, Cloud, and High-Performance Computing (HPC) layers. It approaches the development, the integration planning, and the processes that underpin our two use cases: PER and TASKA from orchestration's perspective.

The Compute Continuum's core objective is to create a dynamic computing environment where tasks are efficiently allocated and executed within the most appropriate computing device or resource, balancing the load and optimizing performance. By addressing specific architectural needs of PER and TASKA, EXTRACT demonstrates versatility and adaptability, ensuring that diverse requirements and different applications are met with efficiency.

To that extent, this deliverable (D4.2) encompasses the journey from initial specifications as described in its predecessor deliverable (D4.1) to the architecture updates, highlighting the evolution of requirements as shaped by technological advancements and project insights during these last 9 months. The document clarifies the Compute Continuum concept and maps out the progression of the implementation, showcasing our achievements and the partial milestones we have set for the upcoming months. In that line, this document presents the technologies and components used to build the compute continuum, without discussing use-case-specific adaptations, which are already explained in D1.2.

The final integration plan outlined in this deliverable represents the development and quality assurance processes. Our integration strategy, consisting of a clear work-flow and conversioning approach, is designed to guarantee interoperability, manage specific configurations when required, and uphold the standards of security compliance and data privacy as per specified in the project.

2. Introduction

The concept of the compute continuum represents a paradigm shift in how we approach the computing infrastructure, bridging the gap between traditional centralized data centres and distributed computing environments. It extends beyond individual architectures, offering a seamless operational layer across edge devices, cloud platforms, and high-performance computing (HPC) systems. This continuum enables data and computational tasks to flow efficiently where they are most logically or physically appropriate, enhancing performance and scalability.

This document, as part of the ongoing effort to define and implement the compute continuum within EXTRACT, focuses on the current state of development at month 15, where we present the status and the current results of the first tangible implementations. It also outlines the progression from initial specifications as per described in the previous deliverable up to the deployment of a Minimum Viable Product (MVP). This MVP is not an end goal but a milestone, showcasing our ability to translate the compute continuum concept into a simple but functional prototype that meets our interim objectives while laying the groundwork for more comprehensive solutions in the upcoming months.

In it, we outline the compute continuum's architecture and its components in a general manner. While we discuss the infrastructure and technologies that form the backbone of the continuum, specific details tailored to our two use cases are covered in deliverable D1.2. There, we dive into how the continuum's architecture is customized to address the unique challenges and requirements of each use case, showcasing its versatility.

Finally, we outline the integration plan of the compute continuum platform with the rest of components and elements of the project.

2.1. Purpose and objectives

The primary objective of this document is to define the architecture and components that will be part of it to establish EXTRACT’s compute continuum platform. In addition, delineate the advancements made towards establishing this foundational compute continuum infrastructure, clarifying the development and deployment of our MVP. This MVP embodies the initial realization of the compute continuum's potential, demonstrating practical applications and benefits while acknowledging its current limitations.

Key purposes and objectives include:

- **Showcasing Progress**: Highlight the concrete steps taken from the project's previous milestone in M6 to reach the current state of the MVP, detailing the technological, architectural, and operational advancements made.
- **Defining Interim Results**: Clearly distinguish between the ultimate goals of the compute continuum infrastructure and the partial achievements represented by the MVP. This involves outlining the specific functionalities, performance metrics, and capabilities enabled by the MVP, as well as the gaps or areas for further development.

- **Setting the Stage for Future Development**: Use the MVP as a benchmark for future iterations, identifying both the successes and shortcomings of the current approach. This sets clear expectations for the project's next phases, including enhancements, expansion of capabilities, and integration of additional components into the continuum.
- **Aligning with Functional and non-Functional Objectives**: Ensure that the progress and lessons learned from the MVP are aligned with the project's overarching goals. This includes improving efficiency, reducing latency, enhancing data processing capabilities, and fostering a more adaptable and resilient computing infrastructure, while ensuring that we address non-functional requirements such as security matters.

2.2. Relationship with other WPs and Previous Deliverables

Deliverable	Tasks	Relation
D1.2	T1.2	First release of the EXTRACT use-cases
D2.2	T2.2, T2.3, T2.4	First release of the EXTRACT data infrastructure and data mining framework
D3.2	T3.2	First release of the data-driven orchestration and monitoring
D4.1	T4.1	Compute Continuum Specification and First Integration Plan

Table 1. Relationship with other WPs

2.3. Document Structure

This document is organized in 5 sections:

- Section 1 is the executive summary of the document.
- Section 2 introduces the document and its objectives.
- Section 3 starts revisiting the requirements set in D4.1 and provides updates on the architecture of our continuum platform, ultimately describing the MVP for WP4.
- Section 4 describes how we are integrating the compute continuum platform with the rest of the elements and the next steps in that line.
- Section 5 ends the document by providing some conclusions.

The document concludes by listing the acronyms, abbreviations and bibliography references.

3. Compute Continuum Requirements

In this section we revisit the compute continuum's requirements identified in Deliverable D4.1 and explain the implementation progression up to date. The objective is to delineate how the initial requirements have guided the development of our computing infrastructure and advancements thus far. Additionally, we address any challenges encountered or adjustments made to out initial approach and the implications these may have.

3.1. Edge

The architecture integrates key components designed to offer robust runtime and management capabilities tailored for edge computing scenarios. These include:

- **Edge SaaS/PaaS**: A cloud-based platform providing centralized management capabilities for edge devices and applications.
- **Edge Agent**: A software agent on edge devices for device management, application life-cycle operations, and secure platform communication.
- **Container Orchestration Engine (COE) and Container Runtime (CR)**: Tools facilitating the deployment and management of containerized applications on edge devices.
- **Operating System (OS) and Edge Device Hardware**: The foundational software and hardware enabling the execution of applications and management agents, optimized for edge computing requirements.

No relevant changes in requirements have been identified since M6.

3.2. Cloud

The cloud requirements identified in D4.1 included:

- **Dynamic VM Provisioning**: A framework for automatic VM provisioning in the cloud to meet specific workload requirements, incorporating auto-scaling for handling peak demands.
- **Deployment System**: A system for deploying workloads to public or private clouds, capable of managing multi-cloud and multi-region deployments with optimization strategies for efficiency.
- **Storage System**: The need for both public and private object storage solutions to support big data needs, with examples including Amazon S3 and CEPH installed on-premises.
- **Container Orchestration Engine (COE)**: A COE like Kubernetes (K8s) is vital for managing and orchestrating containerized applications, ensuring flexibility and portability across different environments.
- **Unified Access Point**: A single access point for managing multi-cluster deployments across hybrid clouds.
- **Monitoring**: Essential for overseeing executions, provisions, components, and identifying failures to efficiently manage big data workloads.

No significant requirements have been added or removed since M6. However, as discussed in Section 4.1, the above requirements have led to an important decision.

3.3. HPC

In the preceding deliverable, D4.1, we identified a set of critical requirements for integrating HPC within the compute continuum to harness its massive parallel processing capabilities and acceleration features, such as GPUs and FPGAs. The key aspects were focused on overcoming HPC's limitations in real-time operations and energy efficiency when dealing with geographically dispersed data sources and large data volumes.

- **Processing Power and Parallelization**: Essential for handling large datasets, necessitating high-performance CPUs and GPUs. Emphasis was placed on efficient resource utilization through parallel programming models like OpenMP, CUDA, and more specifically, COMPSs for data-intensive applications. This approach aimed at optimizing resource allocation by incorporating data-aware scheduling to maximize HPC performance while ensuring software scalability across multiple nodes.
- **Network Bandwidth and Connectivity**: Critical for linking HPC systems with edge and cloud components seamlessly, ensuring fast, reliable data transfer essential for applications requiring quick turnaround times, such as emergency response scenarios. This necessitated high-bandwidth, low-latency networks, coupled with edge data processing to minimize transfer volumes and prevent bottlenecks.
- **Security**: Given the sensitive nature of data processed by HPC systems, stringent security measures were underscored to protect data integrity and confidentiality. This encompassed comprehensive strategies spanning authentication, secure communication, data encryption, and disaster recovery to safeguard against unauthorized access and cyber threats.
- **Software and Tools**: The HPC infrastructure was required to be equipped with appropriate software tools and frameworks to support varied processing needs, including machine learning libraries and data modeling frameworks. Container technologies like Singularity were recommended to enhance software portability and security, ensuring the integrity of software and execution environments.

In the HPC, no changes in requirements have been reconsidered either.

3.4. Security

There is no change on security requirements. For the current state of security architecture and implementation, see corresponding subsection in Section 4 below.

3.5. General Non-Functional

The non-functional requirements for EXTRACT, while particularly critical for edge computing, are also applicable across the entire compute continuum. The requirements we identified emphasize the importance of efficiency, scalability,

reliability, and resource optimization not just at the edge, but throughout the computing spectrum.

- **Minimal (Energy) Footprint**: Ensuring a minimal footprint for the Container Orchestration Engine (COE) and Container Runtime (CR) is essential across all platforms.
- **Performance and Scalability**: High performance, characterized by low latency and rapid response times, is crucial for the seamless execution of applications regardless of their deployment environment. The ability to scale efficiently is equally important, allowing for the seamless integration of an increasing number of devices and applications without performance degradation.
- **Reliability and Availability**: Maintaining high reliability ensures that systems remain operational and accessible. Implementing robust mechanisms to mitigate device failures, network interruptions, and other disruptions is key to minimizing downtime and ensuring continuous operation across the compute continuum.
- **Resource Efficiency**: Optimizing the use of resources by employing smart resource management techniques, such as load balancing and dynamic resource allocation.
- **Security** is as well another key non-functional requirement. As different parts of the infrastructure require different security measures, a more in-depth descriptions has been provided in section 83.4.

We stick to the same non-functional requirements stated in Deliverable D4.1, although we extend those from the edge to the entire compute continuum.

4. Compute Continuum Platform

This subsection outlines the developments until month 15 and any necessary adjustments to the initial compute continuum requirements. We define a new concept we are using in EXTRACT, namely a Compute Continuum Site (CCS).

Besides, we highlight was has evolved from its conceptual design to actual implementation, including the integration of new or replaced technologies in some cases, to ensure that the final architecture effectively meets both the functional and non-functional goals we set. For each component involved in the compute continuum platform, when applicable, we provide the configuration files or developed code to adapt it and be used in the entire platform.

4.1. Public Cloud Provider

Commercial clouds provide diverse eco-systems offering a multitude of resource and service provisioning options. Some of the common offerings include clusters of VMs (Virtual Private Cloud: VPCs) and Kubernetes clusters (K8s), storage - both S3 and block (virtual disks), on-premise connectivity, single point of management (per region) and integrated monitoring. In that aspect, almost all commercial cloud providers can

accommodate the cloud requirements - which further justifies the generality of EXTRACT architecture.

While no particular changes in terms of requirements have been identified, we have started testing the cloud services offered by OVH [12]. Our selection of a public cloud provider is a crucial step towards realizing the continuum infrastructure. After considering several alternatives, OVH has been chosen as our primary provider for cloud services. This decision is based on a comprehensive evaluation of their support, pricing and offerings, which meet the specific needs and requirements of EXTRACT.

OVH stands out as an European alternative, with several location to choose from for their premises, for its robust and scalable cloud solutions that can serve a wide range of computational and storage demands. Their services are designed to support high-performance computing tasks, extensive data storage, and efficient data management - all of which are fundamental to the operations of the continuum.

The choice of OVH allows us to leverage their advanced cloud infrastructure, including virtual machines or Kubernetes instances with varied computational power, auto-scaling them when needed, dedicated GPUs for intensive compute tasks, and flexible object storage options. This infrastructure facilitates the deployment and execution of our compute-intensive workflows, as well as the management and storage of large datasets generated and utilized by our use cases.

As we proceed with the implementation phase, we have initiated a series of tests on OVH's cloud services to assess their performance, scalability, and compatibility with our project's requirements and selected technologies. These tests are aimed at ensuring that the selected cloud platform can effectively support the use cases of EXTRACT, from data processing and analysis to machine learning and simulation tasks. Further details of those tests can be found in Section 216.

4.2. Edge

In Section 3.6 of Deliverable D4.1, we evaluated various edge orchestration solutions, weighing their advantages and disadvantages to identify the most suitable options for the described requirements. KubeEdge and NuvlaEdge were considered the best candidates in that regard. However, as we delved deeper into the integration aspects within the OVH cloud environment, we recognized the necessity for a more cohesive and streamlined orchestration approach. Consequently, we have started testing the use of Kubernetes across the edge as well.

This decision is influenced by the seamless integration and operational efficiency that Kubernetes offers, particularly within the context of OVH's infrastructure. By standardizing on Kubernetes for edge orchestration, we can take full advantage of powerful resources, ensuring not only that they are well-integrated within the overall cloud infrastructure but also capable of meeting the high demands of our compute continuum.

In that sense, adopting Kubernetes across both cloud and edge environments simplifies the management and deployment of applications, providing a uniform operational model that enhances our platform's scalability and flexibility. This unified

approach facilitates a smoother integration with OVH's services from the rest of premises.

Ansible is selected as provision tool (Creation of the K8s cluster). The K8s cluster will be considered as a NuvlaEdge device.

The Nuvla edge and container management platform is used to deploy applications on the created K8s cluster.

4.3. HPC

OVH provides the following dedicated computing instances designed to meet the versatile demands of HPC environments [13]:

- **General Purpose:** These instances offer balanced resources (CPU, memory, storage, and network) for a wide array of production workloads, ensuring high performance for a variety of applications.
- **Compute Optimized:** Tailored for compute-intensive tasks such as scientific simulations and data analyses, these instances facilitate time-sensitive computations with superior efficiency.
- **Memory Optimized:** Ideal for applications requiring high-speed, memory-intensive processing, these instances enhance performance for databases and other RAM-heavy workloads.
- **GPU:** Integrating NVIDIA GPUs, these instances excel in parallel processing tasks, making them well-suited for machine learning, deep learning, and other GPU-accelerated applications.
- **Storage Optimized:** IOPS instances deliver the fastest disk transactions in the public cloud range. They offer direct access to NVMe drives, each of which deliver at least 400,000 read/write operations per second. These cloud solutions are designed to host database servers and big data applications.
- **Discovery:** Discovery instances are based on shared resources, and offer the full experience and features of the public cloud at a lower cost. One can manage their instances with the same tools for their tests, development phases and sandbox environments.
- **Metal Instances:** perfect for workloads that require direct access to compute, storage and network resources. They can also be used to meet requirements for performance, licensing, or customising low-level software layers. They are also compatible with the public cloud ecosystem, enabling the automation of deployment using Infrastructure-as-Code tools.

Counting on the above set of types of compute instances, we can cover a wide range of applications in the HPC, conforming a dedicated CCS able to handle high-compute demanding tasks.

The aforementioned resources, provide a perfect coverage to the processing power needs, the required network bandwidth and connectivity in the HPC, as well as all the necessary tools and security measures (either by providing them or allowing its deployment) as per specified in the detected HPC requirements.

This diverse offering from OVH allows us to establish dedicated CCSs capable of accommodating a broad spectrum of HPC applications. By leveraging these specialized instances, we can ensure that our compute continuum infrastructure is fully equipped

to handle tasks requiring significant computational resources, from routine production workloads to the most demanding scientific computations and data analyses. However, it is important to emphasize that while OVH's resources present a viable option for establishing a robust HPC environment, our selection is not confined to any single provider's offerings. The choice of HPC resources remains flexible, allowing for the exploration and integration of alternative or complementary solutions as deemed suitable. This approach ensures that we remain adaptable to evolving requirements and are able to leverage the best possible technologies to fulfill our objectives. Whether we ultimately decide to utilize OVH's capabilities or opt for other facilities will be determined based on a comprehensive evaluation of performance, cost, and strategic alignment with the project's goals.

4.4. Multi-Cluster Solution

EXTRACT is designed to accommodate the highly diverse constraints of the compute continuum, involving different premises. Edge premises are compute resources that are typically close to, or embedded in, the real-world environment. Thus, edge premises are typically resource-constrained. Cloud and HPC premises are typically far less constrained (can greatly scale), but reside in distant locations, especially in terms of network distance (WAN rather than LAN). The EXTRACT compute continuum need to be able to pool the resources at each premise to make them available to applications on one hand, and to provide a uniform environment for applications and services on the other hand. This lead us to define the EXTRACT compute continuum as a set of clusters with an inter-connecting back-bone that consists of several key services:

1. Orchestration: an EXTRACT application consists of one or more workflows. These workflows may span resources in different premises, regardless of where the application is submitted for execution. Thus, EXTRACT facilitates cross-cluster orchestration using its COMPSs component.
2. Global object storage: the main storage means for data in EXTRACT is S3-compatible object storage. However, object storage is hosted in specific clusters / cloud regions. Through its Sky Store component, EXTRACT delivers a cross-cluster global object store where all buckets and objects reside in a single name space, thus allowing workflows to address object data being used with no regards to the cluster it is hosted in.
3. Data catalog: data in EXTRACT may reside in different locations, services and formats beyond object storage, such as time-series databases, files etc. This requires a global catalog that can be used to locate a specific dataset. Other much-needed capabilities include searching for datasets based on attributes (e.g., author, creation date, format etc), and receiving when a dataset is updated. The EXTRACT Nuvla Data Catalog delivers those functionalities.

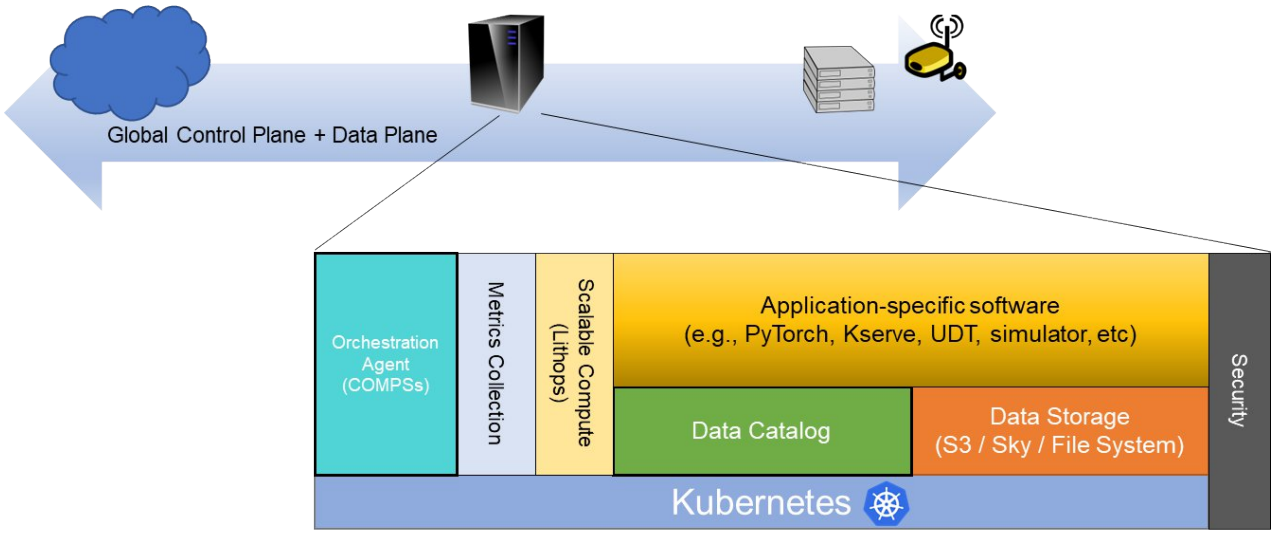

Figure 1 EXTRACT Compute Continuum Site (CCS)

4.5. Compute Continuum Site

As explained above, the EXTRACT compute continuum consists of a set of inter-connected clusters. Each of these clusters is termed a Compute Continuum Site (CCS). A CCS is essentially a Kubernetes (K8s) cluster with EXTRACT component in it, as shown in Figure 1 above. The choice of K8s has been discussed in previous EXTRACT documents, but here we mention several key reasons for it being the cornerstone of the compute continuum:

1. Elasticity: K8s distributions can be deployed on platforms with as little as 2GB ram and 2 cores [minikube], yet they deliver the full K8s API. On large platforms, K8s can scale to thousands of pods / containers. This makes K8s a good fit for a large range of resource-constrained environments.
2. Security: K8s supports a fine-grained security policy such as Role-based Access Control (RBAC), with control over roles and bindings, built-in support for secure protocols and ciphers and additional means. As security is one of mandates of EXTRACT, it's an excellent fit.
3. Popularity: K8s is the modern go-to platform for clusters of containers, used in numerous deployments. It has become near-standard, with familiar tooling such as kubectl and the k8s console. It enjoys a huge catalogued eco-system of the CNCF (Cloud Native Compute Foundation). These properties make it easy to deploy, manage and exploit EXTRACT CCS by users.

On top of the K8s cluster in a CCS there are several EXTRCT components. All the components except the application-specific ones are common - deployed and available in every CCS. The application-specific components may be added depending on the application. In the next sub-sections we discuss each component, referring to additional documentation if required.

4.5.1. Security

For the MVP, the following cybersecurity and data security research and consideration have been explored. These security techniques and practices serve as initial building blocks for both "horizontal security" (i.e., minimally required elements and/or common security elements to all platform's components, such as HTTPS, SSH, and lack of clear-

text protocols such as HTTP, FTP, Telnet; continuous scanning pipelines) and "vertical security" (i.e., increasing the hardening levels of particular security element, such as increasing HTTPS RSA key-lengths from 2048 to 4096+ bits, etc.) cybersecurity build-up.

- **Secure Communications (secure data-in-transit) and Data Security (data-at-rest)**: EXTRACT architecture combines a high number of varied components (either local or distributed) that have communications needs in terms of data planes, control planes, and orchestration. Therefore, one of the main security requirements is to ensure that the communications occur in a uniform, interoperable and secure manner, and that the security and integrity of the data is also ensured. Also, it is important that the security of those communications is underpinned by state-of-the-art security parameters and configurations recommended by leading standardization bodies (e.g., NIST).
    - Current HTTPS communicating endpoints in the architecture must support only TLS v1.2+ and only the Modern cipher suits as follows
        - Modern = *["ECDHE-ECDSA-AES128-GCM-SHA256", "ECDHE-ECDSA-CHACHA20-POLY1305", "ECDHE-RSA-AES128-GCM-SHA256", "ECDHE-RSA-CHACHA20-POLY1305", "ECDHE-ECDSA-AES256-GCM-SHA384", "ECDHE-RSA-AES256-GCM-SHA384"]*
        - This list is constantly checked and updated from CloudFlare Cipher suites recommendations [22].
        - Moreover, for the final architecture, release and demos, weak or self-signed certificates will be forcefully prohibited as part of the preliminary deployment and smoke-test scripts, as well as explicitly mentioning in the accompanying documentation such prohibitions of insecure HTTPS.
            - During local development, tests, integrations and demos such less secure configurations may be perfectly feasible as the aim is to progress and iterate fast during the development stages.
        - The following security architecture requirements have been proposed and agreed so far
            - If for some reason Modern cannot be support, need to discuss internally and see which from "Compatible" is OK (NEVER use "Legacy")
            - EXTRACT App on phone (Samsung) must be build/supporting Modern cipher-suite
            - EXTRACT Cloud (OVH) frontend/APIs must be build/supporting Modern cipher-suites
            - EXTRACT Edge(s) software must be build/supporting Modern cipher-suites
            - EXTRACT "HPC workstation / RL Agents training" software must be build/supporting Modern cipher-suites
    - The communicating nodes exchanging critical information (e.g., information that is used for automated-decision making or has a critical impact on the operations of the whole architecture or deployment), must digitally sign those portions of critically important data sections using strong and state-of-the-art cipher suits, at least a 256bit ECDSA key.
        - "ECDSA keys and signatures are shorter than in RSA for the same security level. A 256-bit ECDSA signature has the same security strength like 3072-bit RSA signature" [23]

- Whenever practically possible and applicable, all the data should be stored on encrypted volume files, whether those are native OS filesystems, ObjectStorage native encryption overlay, or encrypted docker volumes.
- Data security for "RL Agent(s) (inference)" considerations
  - Any "model file" must be accompanied with at least its SHA256 (or SHA512) check value (usually in Linux stored as some-file-name.txt.sha256asc or some-file-name.txt.sha512asc)
    - The type of hash, SHA256 or SHA512, MUST NOT be hardcoded and MUST BE a configuration value to allow easier change, easier testing and security-future-proof upgrades
  - Every time when loading any "model file"
    - According to hash type, check if corresponding sha256asc / sha512asc file exists for the given model
    - If sha256asc / sha512asc does not exist, fails to load (empty or bad permissions), or the SHA256/SHA512 value of the model files differs from the value supplied inside its corresponding sha256asc / sha512asc file ↓ DO NOT LOAD, display error/warning, continue/exit gracefully (depending on the criticality of failed-to-load file and the design of the module)
    - NOTE: Because reading/loading large files and computing hash value of large files has performance/time/resource cost, it is advisable to design "read-and-hash-and-checkhash" routines (should be possible in most modern programming languages) that would read the bytestream of a file and compute hash at the same time

- **Continuous Security Assessment for Cloud/OVH**: One of the final goals for the EXTRACT security architecture is to be able to ensure continuous security scanning, monitoring, visibility, hardening/fixing, and recovery. A first immediate step in such a security strategy is to have an automated approach to scanning the deployed and running resources for various types of security issues, such as known source code or binary vulnerabilities, deployment/runtime security misconfigurations (or secure-but-weak configurations), etc. Since for Phase 2 and MVP, the main scenarios and resources deployed and demonstrated to perform EXTRACT workflows are mainly deployed in OVH, we take advantage of the existing OVH's integration of Trivy continuous security assessment tool, therefore we perform first setup and initial trials of continuous-security-scanning ("horizontal security") for OHV-managed resources (such as Managed Kubernetes).

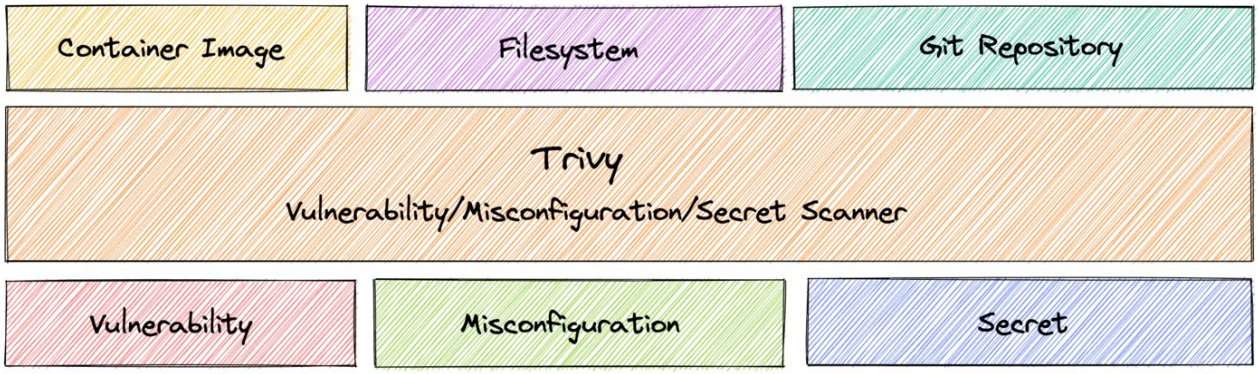

Figure 2: Block-level overview of features and integration options for the Trivy security scanning toolset

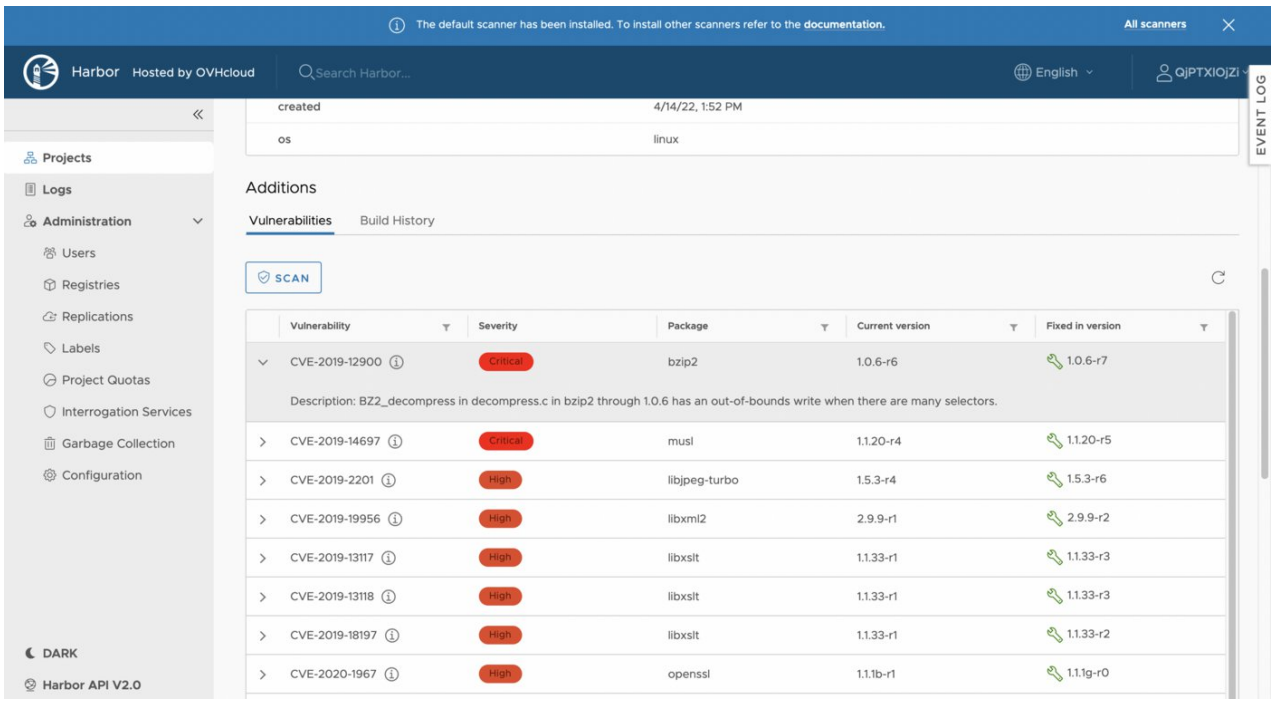

Figure 3 : Example of Trivy security scanner running on an OVH deployed containerized solution

- It is important to note that while we setup and test this Continuous Security Assessment tool in OVH (as the cloud platform selected for MVP and Phase 2), the approach, the automation pipeline and the tool itself (Trivy and similar tools) are not bound to OVH only, and should be in theory easily applicable to other public (paid AWS, Azure) or private (own OpenStack) cloud deployments.

- **Software Supply-Chain and SBOM Mappings**: A complex platform such as EXTRACT is rarely built from scratch and generally is using a relatively big number of external (and complex, potentially vulnerable) libraries, frameworks, technologies, and generic "building blocks" to build, run and interconnect its constituting (and non-trivial) components. Such imported or reused software "building blocks" form what is called the "software supply-chain" of a system or a platform, and those constituting software elements are referenced to as Software Bill of Materials (SBOM) software components [21]. From system

security and security architecture perspectives, the software security part plays one of the most critical parts as the software usually forms the bulk of a software platform and at the same time the software components (especially their complex integrations) are known to introduce bugs and security vulnerabilities. Therefore, the very first step in ensuring software security, hence system security, is to clearly identify (as early in the project as possible, with subsequent continuous followup and updates) all the software components along with their exact version/build numbers (as part of SBOM mapping). Subsequently, the next step is to keep an up-to-date mapping and monitoring of those SBOM software components against databases of known vulnerabilities such as NIST NVD [19]. For flexible and automated queries to such vulnerability database(s), state-of-the-art and/or open-source tools such as FastCVE [20] (developed and maintained by Binare, one of the EXTRACT partners) could be used, where one of its advantages is the flexibility and the ease of its inclusion into Continuous Security Scanning and DevSecOps pipelines.

- As the Phase 2 and MVP architecture and design choices evolved from M6 to M15, the technology contributing partners were surveyed to manually provide the list of SBOM software components they are using for MVP as well as those they envision to use for future developments. Below (Table 1) we provide the list of SBOM software components at the time of this writing, and we expect this list to evolve (both adding new software components as the EXTRACT platform matures, as well as to update versions of already mapped components).

Partner	**Software**	**Version**	Programming language(s)	Is this a third-party	**URLs if avaiable**
BSC	COMPSs Pycompss	/3.0	Python	NO	
BSC	KubeEdge	1.14		YES	https://kubeedge.io/
URV	Lithops		Python	NO	Lithops (lithops-cloud.github.io)
SixSq	Nuvla	-	Clojure, Python	NO	
SixSq	NuvlaEdge	-	Python	NO	
Mathema	Pysyft	-	Python	YES	https://github.com/OpenMined/PySyft
Mathema	CrypTen	-	Python	YES	https://github.com/facebookresearch/CrypTen
Mathema	GeoPrivacy	-	Python	YES	https://github.com/quao627/GeoPrivacy
Ikerlan	Opentelemetry	-	Python	YES	https://github.com/open-telemetry
Ikerlan	Prometheus	-	Config file in .yaml	isNO	https://github.com/prometheus
Ikerlan	Ansible	-	yml	YES	https://github.com/ansible/ansible
Ikerlan	Kubernetes	1.23	yml	YES	https://github.com/kubernetes/kubernetes
IBM	SkyPilot	latest	Python, yaml	Yes	https://github.com/skypilot-org/skypilot
IBM	SkyPlane	latest	Python, yaml	Yes	https://github.com/skyplane-project/skyplane
IBM	ModelMesh	latest	Python, Java, Go, Thrift, ProtoBuf,	No	https://github.com/kserve/modelmesh

			yaml, Go, Bash	
IBM	KServe	latest	Python, Go, Bash, yaml	Yes https://github.com/kserve/kserve
IBM	Caikit	latest	Python, Bash, yaml	No https://github.com/opendatahub-io/caikit
IBM	TGIS	latest	Python, Rust, CUDA, Dockerfile, bash, yaml	No https://github.com/opendatahub-io/text-generation-inference

Table 1: List of selected EXTRACT SBOM software components for use at M15

- Moreover, based on the SBOM information gathered, we performed an initial CVE vulnerability mapping with of those with the following preliminary results below (Table 2). The preliminary results show that some of the components (or specific component versions) used at present **may be** vulnerable to certain attacks or exploits but this must be verified and confirmed first on the specific deployment and configuration, and we aim to perform this for Phase 3 and Phase 4.

Partner	Component	total number of CVE	CVSS version#1	low	medium	high	critical	CVSS version#2	low	medium	high	critical
BSC	COMPSs / Pycompss											
BSC	KubeEdge	8	V2.0	4	3	0	0	V3.1	0	7	1	0
URV	Lithops											
SixSq	Nuvla											
SixSq	NuvlaEdge											
Mathema	Pysyft											
Mathema	CrypTen											
Mathema	GeoPrivacy											
Ikerlan	Opentelemetry	5	V2.0	NA	NA	NA	NA	V3.1	0	1	4	0
Ikerlan	Prometheus	24	V2.0	1	12	2	0	V3.1	0	7	10	1
Ikerlan	Ansible	124	V2.0	50	41	14	0	V3.1	6	54	28	9
Ikerlan	Kubernetes	257	V2.0	27	115	14	0	V3.1	4	104	88	29
IBM	SkyPilot											
IBM	SkyPlane											
IBM	ModelMesh											
IBM	KServe	1	V2.0	0	1	0	0	V3.1	0	0	1	0
IBM	Caikit											
IBM	TGIS											

Table 2: List of selected EXTRACT SBOM software components mapped to their currently known vulnerabilities (if any)

- As immediate steps for Phase 3 in terms of software security architecture, we aim at the following:
  - Evaluation and selection of a industry-best and state-of-the-art static source code analysis tool (such as Whitesource, Checkmarx, Veracode, etc.) able to analyze source code of a large number of programming languages. The selection and evaluation will take into consideration language support (compared to languages used by EXTRACT internal components and its externally dependencies),

pricing, results quality, ease of use and ease of integration into automated and continuous scanning pipelines (DevSecOps style).
        - Run the selected tool on the EXTRACT internal components and its externally dependencies, assess the initial results, understand the limitations of the tools, understand the reported source code vulnerabilities, and suggest remediation for confirmed vulnerabilities.
    - As future steps for Phase 4 in terms of software security architecture:
        - As EXTRACT platform matures towards Phase 4, we aim to integrate the selected source code analysis tool into automated pipelines to be used alongside the "Continuous Security Assessment for Cloud/OVH"

All the current and upcoming elements, techniques and processes of the security architecture related to EXTRACT platform are intended to provide what is known as "Defense in Depth" concept [18], where layering a set of complementary strong security countermeasures build an onion-like or fort-like structure around a system or an asset. This typically means that breaking one layer of security does not allow the attacker(s) to immediately penetrate the entire system but rather it helps keeping the attacker(s) busy trying to break the next layer.

Moreover, even though providing already a strong foundation and understanding of the EXTRACT security baseline and requirements, the above security elements do not exhaustively cover the entire cybersecurity architecture, and we envision that additional security elements and improvements will be added developed and documented for the Phase 3 and Phase 4. Additionally, as nothing comes for free, cybersecurity introduces its own costs (e.g., more complex systems, additional complex computations such as encryption, decryption, hashing, etc.), therefore in Phase 3 and Phase 4 we also aim to evaluate the impact of certain cybersecurity elements on the performance and efficiency of the computations, orchestration and inter-component communications and data-transfers.

4.5.2. COMPSs Orchestrator

Since Kubernetes has become the de-facto platform for deploying and managing containerized applications, several task-based model frameworks (like Ray or Apache Spark, taken into account in D3.1 during the analysis of feasible technologies to be used in EXTRACT) have already been adapted or designed to run efficiently on it. Thus, before adapting the COMPSs framework to be executed on Kubernetes, we conducted a small state-of-the-art study to know how these frameworks have been adapted or designed to run efficiently on it. After the study, we were able to differentiate two approaches:

- Kubernetes-native solutions. This approach leverages Kubernetes Custom Resource Definitions (CDRs) and Custom Controllers. Some examples include the KubeRay Operator[14] and the Dask Kubernetes Operator[15].
- Command line adaptation. This solution expands an existing CLI so the application can also be deployed in Kubernetes. An example is the *spark-submit* CLI[16] that was modified to take new arguments.

Furthermore, we also checked how COMPSs had been adapted to other containerized environments. Before the EXTRACT project, COMPSs was already prepared to be executed in Docker Swarm, using the *runcompss-docker* command line; and in HPC environments, using the *enqueue_compss* command line.

Taking into account the different implementations of task-based frameworks for Kubernetes and the implementation of COMPSs in other containerized environments, the decision for EXTRACT has been to create a new command line, *runcompss-k8s*, that allows to execute COMPSs workflows in the containers orchestrator.

The *runcompss-k8s* allows to execute a COMPSs application using its master/worker paradigm. Under this paradigm, the COMPSs master is in charge of starting the application and offloading the computing tasks to the workers. As per the workers, they must have their SSH port opened and be ready to receive the tasks sent by the master.

The new command line deploys both, master and workers, as Pods and creates a headless Service that allows all the Pods to communicate with each other through DNS. Furthermore, a Kubernetes Cloud Connector has been developed, which allows for elasticity during workflows executions, allowing the COMPSs runtime to create new worker Pods in the middle of an execution if necessary.

4.5.3. Storage

EXTRACT's storage componentry is designed to enable EXTRACT application components that are deployed in the current CCS to access EXTRACT data that is stored in file-systems and object storage. File-system access is delegated to K8s' volume mounting in containers. For object storage, EXTRACT provides universal access to all objects, regardless of their location (in the current CCS/cloud region or not). This is because object storage is regarded as the primary storage means for EXTRACT data. Regardless, all datasets in EXTRACT are tracked through the data catalog and can be located and remotely accessed - see subsection 4.5.5.

We now focus on the global object store, namely the Sky Store component. Sky Store is a facility for providing unified global efficient access to object storage, regardless of its location - in the same cluster or cloud region or not.

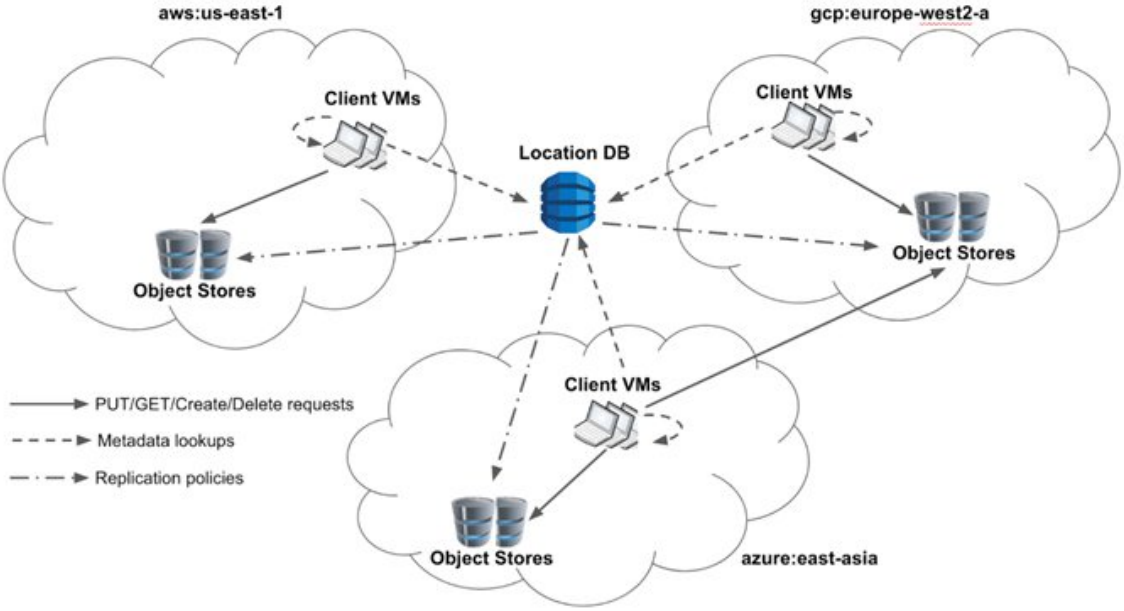

Figure 4 Sky Store deployed across multiple premises

As can be seen in Figure 4 above, a Sky Store deployment consists of a local client (i.e., in every CCS) and a location database, which implements the global namespace. The location DB can be replicated for scale and resiliency. It needs to be deployed in only one or few CCS. The local client implements the same S3 protocol that the concrete object storage systems implement, but it serves as a *proxy*, in the sense that accessing an object or bucket through the client may be delegated to the local object storage or a remote one.

Recall that Sky Store provides a single global namespace for all the buckets (and hence objects in buckets) used by EXTRACT. This is achieve by a mapping from the global names of buckets to the name and location of concrete buckets in various clusters / regions / premises. That mapping is stored in the location DB. When an application accesses a global bucket through a Sky Store client, the first operation of the client is to access the location DB to retrieve the mapping, if it's not already in the client. Then, the client establishes itself as a proxy to the actual (mapped) bucket.

This architecture is simple and flexible, and in particular it allows incorporating policies for controlling replication and consistency of the stored data. One common feature that is used in EXTRACT is local caching - when a client accesses a remote bucket and there is local object storage available, it may cache the remote bucket locally to improve read access. At the application level, as discussed above, this allows EXTRACT applications / workflows to address their object data conveniently, regardless of its location.

4.5.4. Data Staging with Lithops

Lithops, Extract's main component for data staging, can leverage the compute continuum to perform data staging across distributed environments by adopting a serverless computing model. As described in D2.1, Lithops provides an abstraction layer where users only code simple functions to process collections of data. It can then deploy and run these functions at scale on top of different infrastructures, from cloud-

specific services like AWS Lambda or EC2, to on-premises or edge devices through resource management tools like Kubernetes, allowing for seamless data processing and movement. The tool offers different compute backend implementations to seamlessly run functions atop the different services and technologies.

Lithops can easily plug into Extract's Compute Continuum Sites (CCS) by using the appropriate backend. In particular, for Extract, Lithops is configured to use Kubernetes as a backend for computation, which deploys containerised functions as pods within the cluster. This allows Lithops to leverage the compute continuum management of Extract to dynamically allocate resources across it. For data storage, Lithops has multiple storage backends available. For Extract, we configure it to use the S3-compatible object storage backend, which will enable it to directly access Extract's data layer, backed by any cloud storage solution that offers an S3 API.

Beyond this interconnection of the data staging layer with the compute (CCS) and storage (data layer) resources, logically Lithops also integrates simply with the orchestration layer. As described in D3.1 and planned in D4.1, the main goal is to utilise COMPSs as the global orchestrator, which will focus on scheduling complex computational workflows and manage the parallel tasks that compose them efficiently across the distributed resources. To perform data staging operations, COMPSs may use Lithops and its data-driven smart provisioning to effectively scale the data-intensive operations in a workflow. Lithops will become a second-layer orchestrator of those operations. In some cases, the opposite may also happen. Lithops may use COMPSs in a particular task to perform certain distributed compute-intensive logic. As a general rule, COMPSs takes care of the overall orchestration of workflows and focuses on the compute-intensive parts of data mining, while Lithops will be in charge of performing data-intensive operations in the data staging section of a workflow.

4.5.5. MetaDataCatalog

As described in section 3.1 of D2.1, the metadata Catalog catalog leverages the positive attributes of S3-based services and introduces a comprehensive global management system for metadata. The goal is to enhance the efficiency of search functionalities across different service providers. In terms of implementation, the model consists of three core resources

1. data-object: This resource acts as a proxy for data stored in an S3 bucket/object from a specific provider. It manages the lifecycle of S3 objects, simplifying data upload and download processes.
2. data-record: This resource allows users to add additional, user-specified metadata for an object. Enabling the attachment of rich, domain-specific metadata to objects enhances the precision of searching for relevant data.
3. data-set: This resource defines dynamic collections of data-object and/or data-record resources through filters. Administrators, managers, or users can define these collections, providing a flexible and customizable approach to data organization.

Collectively, these resources establish a versatile data management framework applicable to EXTRACT use-cases. The typical workflow involves creating a data-object (implicitly creating the S3 object), optionally adding metadata using a data-record object, and finally, finding and using the relevant data-object resources included in a data set. Nuvla facilitates the "using" element by binding data types to user

applications capable of processing the data, offering seamless integration between data management and application utilization.

4.5.6. Monitoring Layer

The monitoring layer oversees the whole Extract architecture. This includes both the infrastructure and the workload. Multiple daemons will be deployed among all the cluster nodes which will collect the different monitored metrics and send them to a centralized server. All this metrics will be made available to both the COMPSs orchestrator as well as other visualization tools.

Prometheus has been used to implement this layer. It provides EXTRACT with a centralized monitoring service which will periodically retrieve the measured data from the different daemons. These daemons have been implemented as native Prometheus exporters, as well as using other tools such as OpenTelemetry. Grafana has been selected as the visualization interface for these metrics. A more detailed description of the monitoring layer can be found in deliverable D3.2, including a complete list of metrics that are being captured.

5. First Release of the Integration Plan

This section outlines the integration plan for EXTRACT, building upon the preliminary integration framework outlined in Deliverable D4.1. It encompasses the collaboration and coordination across all work packages, aiming to bring together all components, tools, and technologies that form the foundation of our compute continuum infrastructure.

The following are the five tasks that underpin WP4:

- **T4.1**: Compute continuum requirement specification and EXTRACT platform integration plan (M1:M15)
- **T4.2**: Programming and execution models interoperability (M7:M30)
- **T4.3**: Cybersecurity (M7:M30)
- **T4.4**: Compute continuum validation (M31:M36)
- **T4.5**: EXTRACT software platform prototype integration (M7:M36)

	Year 1		Year 2	Year 3
	Phase 1	Phase 2	Phase 3	Phase 4
WP4				
T4.1				
T4.2				
T4.3				
T4.4				
T4.5				

Table 3: Tasks in WP4

This Section is mainly addressed at finalizing Task 4.1, which has been completed by setting the requirements in Deliverable D4.1 and with the hereby proposed integration plan.

5.1. Overview of the Initial Integration Plan

In D4.1 we described EXTRACT's integration plan. In it we defined the processes to be followed, what defined the infrastructure, and the standards and guidelines that will be followed throughout the project.

No relevant issues or required changes have been identified from M6 to M15, so that plan is still valid.

5.2. Integration Phases

Figure 6 displays the different phases in which we planned to integrate all elements from EXTRACT until producing the desired framework.

The initial phase of the project focused on establishing the requirements for each WP. Right now we are finishing Phase 2, for which an MVP for EXTRACT has been defined and implemented. This MVP, as aforementioned, is aimed at joining all the components within a WP to start the integration at a work package level.

From the architectural point of view, this EXTRACT MVP consists of the following work-package subsets:

WP1: (see D1.2)

- PER use-case: single hazard (fire), short scenario from alarm to first phone instructions (possibly as separate functions)
- TASKA use-case: only TASKA-C sub-use-case, one scenario of single offline dataset processing

WP2: (see D2.2)

- Data lake: data catalog
- Elastic compute ETL Model serving: Serverless Kserve with S3 on Kubernetes
- Model training

WP3 & WP4:

- First CCS in OVH Cloud, Gravelines region, France
- COMPSs running in OVH CCS on K8s using OVH S3

We have implemented the MVP that is further described in 6, which integrates the cloud infrastructure provided by OVH. This MVP, aimed at testing and combining the efforts of Work Packages 3 and 4, is crucial for establishing a preliminary version of our compute continuum infrastructure. Through this MVP, we executed workflows that illustrate our ability to manage, process, and interact with data across a basic yet functional continuum setup. Specifically, it involves reading datasets from object storage, processing these datasets through serial and parallel computations orchestrated by COMPSs, and storing the outcomes back into OVH's cloud storage using the S3 API.

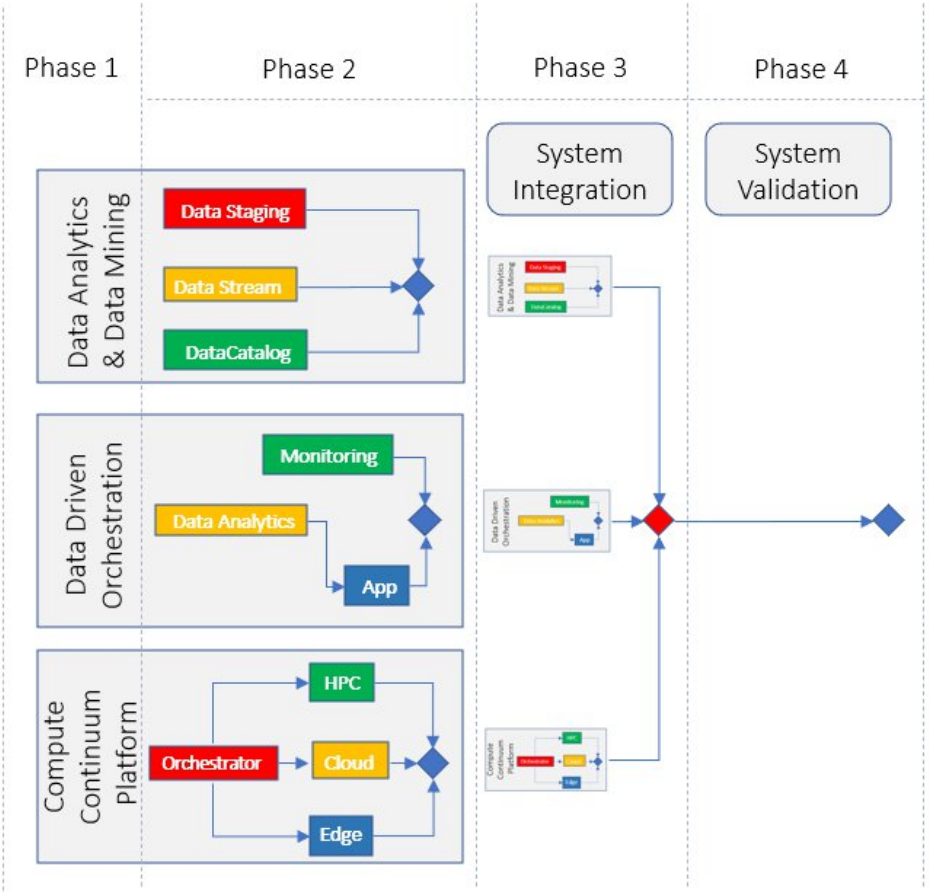

Figure 6: Integration Phases of WPs

Next, in Phase 3, our aim is to evolve from these foundational components established in Phase 2 to deploying the actual use cases that leverage the full capabilities of our compute continuum infrastructure. This phase is dedicated to realizing a comprehensive end-to-end pipeline that encompasses the entire data lifecycle: from initial data ingestion and processing to advanced data mining, and the execution of complex workflows across the most suitable components of the compute continuum.

The key objective during this phase will be to demonstrate the seamless operation of our system by executing tasks where they are optimally suited—whether on the edge

for real-time processing, in the cloud for scalable compute resources, or on HPC for intensive computations.

To achieve this, we will focus on scaling our infrastructure both vertically, by enhancing the capabilities of each component, and horizontally, by integrating more components into the continuum. This will allow us to handle more complex and demanding tasks characteristic of our targeted use cases. The system will be underpinned by a monitoring system, ensuring that we have a comprehensive view of the system's performance, resource utilization, and the execution status of tasks across the continuum.

Finally, in phase 4 we will devote our efforts to test and validate the implemented solution, refining or improving it where possible.

6. Infrastructure MVP

6.1. Demonstrator - COMPSs on K8s

At the current moment of the project, M15, we have created an MVP for our platform, consisting of the above components in order to execute simple workflows. Therefore, this MVP is a foundational step towards constructing the final compute continuum infrastructure by the end of the project. The essence of this MVP is to begin the practical testing of the OVH cloud infrastructure, defining a first sample of a CCS and to incrementally integrate additional components and technologies as the project progresses, enabling us to better comprehend the interactions among them and detect as early as possible any issue or incompatibility.

To that end, this MVP makes use of the following components described in previous sections:

- OVH's Public Cloud Managed Kubernetes, with a pool of 3 small instances (7GB RAM, 2 vCores, 50GB SSD each)
- OVH's Object Storage (10GBs)
- Deployment of COMPSs applications

The test that has been successfully executed, consists on deploying two similar workflows that demonstrate the project's capability to manage, process and store/retrieve objects across this minimal CCS. This is achieved through two Python applications that are designed to read two datasets from an object storage bucket, execute a range of computations—both serial and parallel, with the objective of creating workflows as in Figure 7, which are distributed among the different nodes we count on—and then store the results back into OVH's cloud storage using the S3 API.

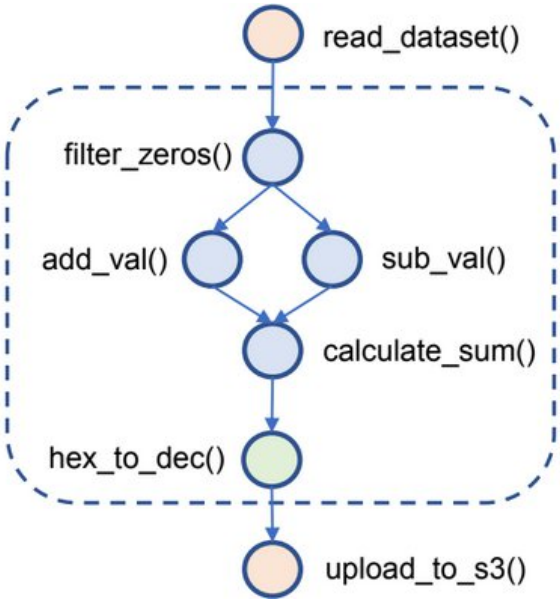

Figure 7: COMPSs MVP Workflow

To create the aforementioned workflows, we employ COMPSs as the application-level orchestrator.

This simple test aimed at starting having small but functional applications running in the infrastructure, interacting with several of the needed components, such as the S3 API or OVH's cloud. Furthermore, understand if the premises offered by the selected cloud provider met our needs in all terms.

Going into a bit more detail, the two applications start by reading their input datasets, stored in OVH's object storage bucket. These, are named "img01.hex" and "img02.hex" respectively (Figure 8).

Name	Latest modification	Size
test1.txt	20 Feb 2024 10:50:47	17 B
img02.hex	20 Feb 2024 14:22:44	319 B
img01.hex	20 Feb 2024 14:22:45	239 B

Figure 8: COMPSs MVP input datasets

The content of the datasets is a very trimmed representation of an imaging dataset from TASKA, consisting of hexadecimal values, such as:

bebe bebe 0000 0e26 0000 0005 5461 626c

6500 0000 0200 0000 5100 0000 0100 0000

0a50 6c61 696e 5461 626c 6500 000b e200

0000 0954 6162 6c65 4465 7363 0000 0002

0000 0000 0000 0000 0000 0000 0000 0035

0000 000b 5461 626c 6552 6563 6f72 6400

After the object is retrieved from the bucket by the application, its first task consists of filtering all the zeroes. Afterwards, a copy of the resulting dataset is created and stored in a data structure, and in one of the copies a value is summed to every single hextet, while another value is subtractet to each hextet of the other copied dataset. This generates two individual tasks that can be performed in parallel without any dependencies, and hence, be potentially executed by different nodes concurrently. Once both tasks have finished (creating a data dependency), both outputs are summed into a 3rd resulting dataset. As the latest step, each hextet of the resulting dataset is converted to decimal values, and both results (the hexadecimal one and the decimal one) are stored back to the same object storage bucket, as shown in Figure 9.

Figure 9: Workflow 1 Outputs

The same exact process is identically performed by the 2nd workflow. The reason behind having two separate workflows performing the same steps, is to further test in the upcoming months that we can distribute workflows in different clusters (multi-cluster orchestration), although at this moment, both were executed in the same one.

Finally, a 3rd workflow retrieves both hexadecimal results (and hence it needs to wait for both workflow1 and workflow2 to have finished) and simply joins them.

As it can be noted, the operations do not aim to have any logical meaning, as this is not the objective of this MVP.

This integration is therefore meant to pave the way for future expansions and enhancements of the compute continuum infrastructure. In that line, the main goal for the upcoming months will be to keep expanding our current infrastructure both vertically and horizontally. That is, incorporating the components and technologies described in this document while also using more powerful instances that can handle more demanding tasks.

We have uploaded the code of these applications and a more detailed guide of the steps to be followed in order to properly prepare the environment as well as the instructions to be executed in our EXTRACT Gitlab repository. Concretely, this readme file contains the instructions followed.

6.2. Demonstrator – Sky Store

Sky Store is the intended backbone for delivering global object storage (S3) access for all EXTRACT components and applications, as discussed in Section 4.5.3. For MVP purposes, we have created a demonstration (using already-working components) of global data access between two geographically-distinct cloud regions. This demonstrator includes first a description of the demo story with instructions and technical explanations where needed. Then, there is a video showing the actual demo.

Last, we provide a repository for installing the code based on the instructions in the demo story so that you can experience the demonstration on your own.

6.2.1. Demo Story

This demonstration is about global object storage access using Sky Store. The actual data is stored in two distinct AWS regions:eu-central-1 (Frankfurt) and eu-west-1 (Ireland). This demo does not use OVH yet, nor the geographical regions selected for EXTRACT. Rather, it's using IBM-provisioned resources related to the current development work.

In the demo setup, we create two computing environments, each connected to a separate object storage cluster. These computing environment simulate two EXTRACT CCS. We then show that each of these environments can create buckets and objects in a virtual name space provided by Sky Store, such that the other environment sees the same global space and can access it.

Both environments are Linux instances. One is a personal laptop and the other is a VM. In accordance with Section 4.5.3, in each environment we deploy the Sky Store proxy client `s3-proxy` . We also need a publicly-accessible location database server, which is the `skystore-server`. For the demo, we deploy it by arbitrary choice in the VM. The server prototype does not support yet a secure connection, so in order to connect the client proxies with the server we also set up an SSH tunnel for the server port 3000 between the laptop and VM. Figure 10 below details the demo setup.

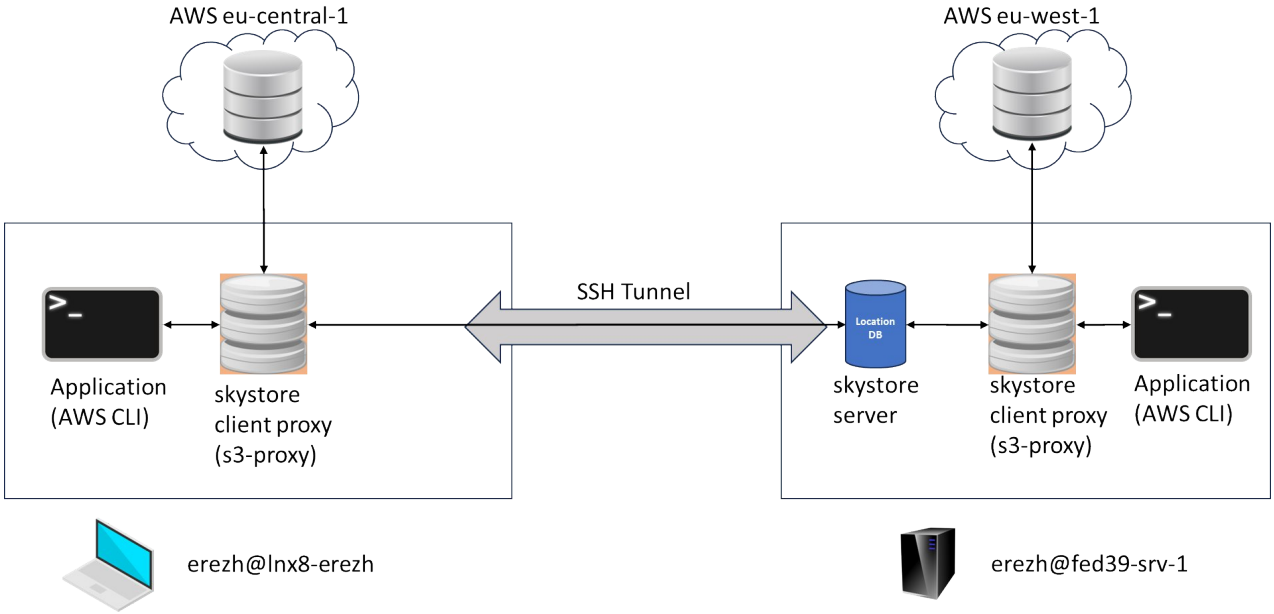

Figure 10 Sky Store Demo Layout

For the demo itself, we begin by listing the buckets in Sky Store from either environment, showing that there are currently no buckets or objects. We then create a bucket in one env, and list buckets in the other, showing that the bucket is globally accessible. Last, we use one env to put an object in a bucket create in the other env, and then read the object to the other env. Last, we present the real buckets created in AWS object storage by Sky Store. These steps demonstrate that Sky Store facilitate a global object store with a single name space that is rooted in actual object storage.

6.2.2. Demo Execution

1. **Environments.** Prepare two separate Linux environments, *env1* and *env2*. Each env can be a machine (BM/VM), container, etc - should be isolated from the other env. Need to have a user *u1@env1* and *u2@env2*. No sudo is required.

2. **Setup.** In each env, you need to prepare:

    1. **Python tooling with Python virtual environment**. We recommend to install `pyenv` as a way to do this. Then, we create a Python 3.12.1 virtual env called `skystore-test` using the following commands:
    ```
    pyenv install 3.12.1
    pyenv virtualenv 3.12.1 skystore-test
    pyenv activate skystore-test
    ```
    2. **Rust tooling**. Install Rust using the recommended installer:
    ```
    curl https://sh.rustup.rs -sSf | sh
    source "$HOME/.cargo/env"
    cargo install just —force
    ```
    3. **AWS credentials and tooling**. You need to obtain AWS credentials. You can use the same credentials for both users. Basically, you need to have a script, called e.g., `env_aws.sh` that sets up the credentials for each env, like this:
    ```
    export AWS_ACCESS_KEY_ID="XXXXX"
    export AWS_SECRET_ACCESS_KEY="XXXX"
    ```
    AWS CLI tooling is used for testing the Sky Store S3 access. It can be installed from here.
    4. **Clone and build Sky Store repository**. Just clone the repository, set the current branch and build as following:
    ```
    git clone https://github.com/gilv/skystore
    cd skystore
    git checkout skystre-server-addr-config
    cd s3-proxy
    cargo build
    cd ../store-server
    pip install -r requirements.txt
    cd ..
    pip install -U .
    ```

3. **Start the Sky Store server** in one of the environments. To do that, go to that environment, e.g., u1@env1, and set up a few environment variables. To make this repeatable, create a script `env_skystore.sh` as following:

```
export INIT_REGIONS=aws:eu-central-1,aws:eu-west-1

export SKYSTORE_BUCKET_PREFIX=<prefix with no white space, e.g., skystore-yourname>
```

Note that `INIT_REGIONS` needs to cover all the premises that will be covered by Sky Store. For now, those are as described in the demo.

Now, start the sky store server (location database) with the following command in the s3-proxy folder of the repository:

```
. <your path to>/env_skystore.sh

just run-skystore-server
```

This will make the skystore server start on port 3000 of the respective host. To make the other environment reach the skystore server as well, you need to set up an SSH tunnel for port 3000 between both envs, by running the command below in the **second env**:

```
ssh –L 3000:localhost:3000 –N –f u1@env1
```

4. **Start the sky store client proxy** in each env. To do that, you first need to create a configuration file (JSON) for each respective client, telling it which real object storage to use and where is the sky store server, as well as data access policies. In one env you will use the object storage of AWS region eu-central-1, so create a config file called `eu-central-1.json` with the following content:

```
{
 "init_regions": [
   "aws:eu-central-1",
   "aws:eu-west-1"
  ],
 "client_from_region": "aws:eu-central-1",
 "skystore_bucket_prefix": "skystore-yourname",
 "policy": "write_local",
 "server_addr": "127.0.0.1"
}
```

Then issue the following commands in the `s3-proxy` folder to start the Sky Store client proxy:

```
. <your path to>/env_aws.sh
skystore init –config=<your path to>/eu-central-1.json
```

The client proxy starts as both a client of the skystore server and the AWS object storage, as well as an S3 server (hence a proxy) listening on port 8002.

In the second env, create a configuration file eu-west-1.json with the same content as the previous JSON above, but change one line as below:

```
 "client_from_region": "aws:eu-west-1",
```

Then similarly start the sky store client proxy in the `s3-proxy` folder:

```
. <your path to>/env_aws.sh
skystore init –config=<your path to>/eu-west-1.json
```

5. Execute the demo itself. Before running the following commands in a shell in each env, be sure to first set the AWS credentials using the command below:

```
. <your path to>/env_aws.sh
```

Using AWS CLI tooling, engage the Sky Store client as any standard S3 service. The following command sequence illustrate the demo scenario and the expected replies:

1. List the current buckets in each of the envs:
   ```
   aws s3api --endpoint-url http://127.0.0.1:8002  --no-verify-ssl
   list-buckets
   ```

The expected reply is an empty list - because currently there are no buckets in the virtual Sky Store space - even if you currently have buckets from other applications in the respective AWS region object storage.

2. Create a bucket in the env with `eu-central-1`:
   ```
   aws s3api --endpoint-url http://127.0.0.1:8002 --no-verify-ssl create-bucket --bucket eu-central-1-bucket-1
   ```
3. List the current buckets in each of the envs again (see command 1 above). You will see BOTH clients list the new bucket eu-central-1-bucket-1. This is because Sky Store is a global unified name space for all data.
4. Create a bucket in the env with `eu-west-1`:
   ```
   aws s3api --endpoint-url http://127.0.0.1:8002 --no-verify-ssl create-bucket --bucket eu-west-1-bucket-1
   ```
5. List the current buckets in both clients again (command 1). You will see each client lists both buckets.
6. Using the client with `eu-central-1`, put an object into the `eu-west-1-bucket-1` bucket. First we create a synthetic payload file of a specified size (5MB in this case) and then we upload it into the bucket:
   ```
   truncate -s 5M payload.data
   aws s3api --endpoint-url http://127.0.0.1:8002 --no-verify-ssl put-object --bucket eu-central-1-bucket-1 --key payload --body "./payload.data"
   ```
7. In both clients, list objects in the bucket `eu-west-1-bucket-1` using the command below. Both clients should show the object `payload` of size 5MB.
   ```
   aws s3api --endpoint-url http://127.0.0.1:8002 --no-verify-ssl list-objects --bucket eu-west-1-bucket-1
   ```
8. Using the client with `eu-west-1`, download the `payload` object from the bucket `eu-west-1-bucket-1` by executing the command below. You should have received a 5MB file. You can later send it to the other env to compare and make sure it is identical to the original file.
   ```
   aws s3api --endpoint-url http://127.0.0.1:8002 --no-verify-ssl get-object --bucket eu-west-1-bucket-1 --key payload payload.data
   ```

6. Now that you've experienced working with the virtual global environment of Sky Store, browse to your AWS object storage console to see the actual buckets and objects created by Sky Store. You should see two buckets whose names begin with skystore-yourname, one in each region of `eu-central-1` and `eu-west-1`. In the one of `eu-west-1` there should be an object called `payload` of size 5MB.

6.2.3. Demo Video

The video linked below delivers the demo story as explained in Section 6.2.1 and detailed in Section 6.2.2. To avoid a lengthy video with mundane details, we skip the whole setup part and present just the demo itself, of engaging with buckets and objects over Sky Store.

https://b2drop.bsc.es/index.php/s/eRQrQ6jHbHGnGkT

6.2.4. Demo Resources

Sky Store repository on Github: https://github.com/gilv/skystore

6.2.5. Next Steps

As shown in this demo, the current MVP prototype is already capable of delivering the core capability of a single virtual globally-accessible object storage. However, there are many smaller features and NFRs we need to address in the following releases to achieve more practical usability. These include object caching, support for non-cloud premises such as Minio or Ceph, secure connections, K8s deployment, and more. In addition, we need to properly evaluate the performance of Sky Store for some key EXTRACT usage patterns.

7. Conclusion

This deliverable marks a significant milestone within the EXTRACT Work Package 4 (WP4). Specifically, it presents the inaugural release of the EXTRACT compute continuum platform and outlines the initial steps of the integration plan. In this Deliverable D4.2, we have provided an update on the current progress of WP4. This includes the inclusion of the concept of the Compute Continuum Site (CCS), a delineation of its components' structure, and an explanation of the interactions and configurations required among them. Additionally, we have executed a preliminary integration through the development of Minimum Viable Products (MVPs). These MVPs are instrumental in laying the groundwork for the full platform construction. They enable early-stage infrastructure testing and assessment of services provided by our chosen cloud partner, OVH.

To demonstrate their operational efficacy, we present two demonstrators.

Furthermore, this document details our advancements in the project's integration plan. We are pleased to report that our progress is aligned with the planned timeline. The document concludes by outlining the subsequent phases and actions slated for the forthcoming months.

8. Acronyms and Abbreviations

- AI – Artificial Intelligence
- API – Application Programming Interface
- AWS – Amazon Web Services
- CA – Consortium Agreement
- CCS – Compute Continuum Site
- CD – Continuous Development
- CI – Continuous Integration
- COE – Container Orchestration Engine
- CPU – Central Processing Unit
- CR – Container Runtime
- D – Deliverable
- DevOps – Development and Operations
- DevSecOps – Development, Security and Operations
- DoA – Description of Action (Annex 1 of the Grant Agreement)
- GA – General Assembly / Grant Agreement
- GPU – Graphics Processing Unit
- HPC – High Performance Computing
- ISO – International Organization for Standardization
- KPI – Key Performance Indicator
- K8s - Kubernetes
- M – Month
- MS – Milestones
- NAT – Network Address Translation
- OS – Operation System
- PaaS – Platform as a Service
- PM – Person month / Project manager
- QUIC – Quick UDP Internet Connections
- SaaS – Software as a Service
- SHA – Secure Hash Algorithm
- T – Task
- TLS – Transport Layer Security
- UDP – User Datagram Protocol
- VCS – Version Control System
- VM – Virtual Machine
- VPN – Virtual Private Network
- WP – Work Package

9. References

[1] https://openmp.org
[2] https://developer.nvidia.com/cuda-zone
[3] http://compss.bsc.es
[4] S. Chang, et.al., *"Feasibility of Running Singularity Containers with Hybrid MPI on NASA High-End Computing Resources"* CANOPIE-HPC, 2021
[5] https://aws.amazon.com/s3
[6] https://www.ovhcloud.com/en-gb/
[7] https://www.redhat.com/es/technologies/storage/ceph
[8] https://kubernetes.io/
[9] https://skypilot.readthedocs.io/en/latest/
[10] https://kubeedge.io/en/
[11] https://nuvla.io
[12] https://www.ovhcloud.com/en-ie/
[13] https://www.ovhcloud.com/en-ie/public-cloud/compute/
[14] https://docs.ray.io/en/releases-2.7.0/cluster/kubernetes/index.html
[15] https://kubernetes.dask.org/en/latest/operator.html
[16] https://spark.apache.org/docs/latest/running-on-kubernetes.html#how-it-works
[17] https://help.ovhcloud.com/csm/en-public-cloud-kubernetes-install-trivy?id=kb_article_view&sysparm_article=KB0049874
[18] https://csrc.nist.gov/glossary/term/defense_in_depth
[19] https://nvd.nist.gov/
[20] https://github.com/binareio/FastCVE
[21] https://www.cisa.gov/sbom
[22] https://developers.cloudflare.com/ssl/reference/cipher-suites/recommendations/
[23] https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages