EXTRACT

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

D4.1 Compute Continuum Specification and first integration plan

Version 1.0

Documentation Information

Contract Number	**101093110**
Project Website	www.extract-project.eu
Contractual Deadline	M6, June 2023
Dissemination Level	PU
Nature	R
Author	IKL
Contributors	BIN, BSC, IBM, IKL, SIX
Reviewer	SIX
Keywords	Continuum, requirements, plan

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

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

Change Log

Version	Description Change
V0.1	Initial Draft
V0.2	Ready for final review
V1.0	Final review

Table of Contents

1. Executive Summary 4
2. Introduction 5
2.1. Purpose and objectives 5
2.2. Relationship with other WPs 5
2.3. Document structure 5
2.4. Compute Continuum overview 6
3. Compute Continuum layer 7
3.1. Interoperability abstraction layer 7
3.2. HPC requirements 7
3.3. Cloud requirements 10
3.4. Edge requirements 11
3.4.1. Functional requirements 13
3.4.2. Non-functional requirements 14
3.5. Security requirements 15
3.5.1. Cybersecurity requirements 15
3.5.2. Data security requirements 16
3.5.3. Security of ML models 17
3.6. Compute Continuum layer high level design 17
4. Initial Integration Plan 23
4.1. Processes 24
4.1.1. Development and Integration Process 24
4.1.2. Quality Assurance Process 26
4.1.3. Unit Testing 26
4.1.4. Regression Tests 26
4.1.5. Bug and Issue Tracking 27
4.2. Infrastructure 27
4.2.1. Development Platform: 27
4.2.2. Instant Messaging and Transparency 28
4.2.3. Integration Platform 28
4.2.4. Quality Assurance Tools 30
4.2.5. File Storage and Collaboration 30
4.3. Standards and Guidelines 31
4.3.1. Design Patterns 31
4.3.2. Code Comments 31

4.3.3. Programming Style 31
5. Conclusion 32
6. Acronyms and Abbreviations 33
7. References 34

1. Executive Summary

This deliverable represents the work completed in the first phase of the project within WP4. It brings together the tasks performed in T4.1, which include defining the requirements for the Compute Continuum and outlining the integration plan for the EXTRACT platform, with the aim of reaching Milestone MS1.

This deliverable focuses on defining the complete Compute Continuum ecosystem, which serves as the foundation for the EXTRACT research activities. It begins with an analysis of the current state of art, providing an overview of the Continuum architecture concept and examining various architectures. Special attention is given to projects and initiatives in which EXTRACT partners have contributed.

To ensure compatibility with the use cases defined by the providers in WP1, the Cloud, Edge and HPC requirements have been analysed. The analysis aims to identify the requirements that have an impact on the selection of hardware platforms and components for the Continuum architecture.

Considering these requirements, a specific set of parallel and low-power embedded hardware architectures has been chosen for the Edge level. These architectures can meet the performance requirements for early analytics while ensuring adherence to non-functional properties.

At the Cloud level, this deliverable defines the Cloud architecture, components and functionalities necessary for conducting in-depth analysis on massive volumes of data.

On the other hand, High-Performance Computing (HPC) technologies are employed to address the volume and speed characteristics. These technologies support parallel processing capabilities, advanced acceleration features and facilitate the training and execution of complex AI models. Such capabilities are crucial for handling data complexity and incompleteness.

Furthermore, the cybersecurity aspects have been specified to ensure the mitigation of data privacy and security risks across all relevant levels. This includes measures to safeguard data integrity and protection, secure Machine Learning models, trusted computation, and secure communication channels. The defined characteristics aim to minimize potential vulnerabilities and ensure the overall security of the system.

Additionally, this also includes the design of a standardized high-level abstraction layer that facilitates the deployment and efficient execution of data mining workflows across the entire Compute Continuum. This layer will leverage and improve existing programming paradigms, execution models and virtualisation techniques employed in Edge, Cloud and High-Performance Computing (HPC), enabling seamless interoperability for processing diverse data with distinct attributes.

The document concludes by presenting an initial proposal for the integration plan of the EXTRACT platform, with a specific focus on closely monitoring the progress and advancements from WP2 to WP4. This integration plan serves as a framework for overseeing the implementation of the platform and ensuring alignment with the research activities conducted in these work packages.

The initial milestone of task T4.1 has been accomplished successfully, with all its objectives met, as documented in this deliverable.

2. Introduction

The primary objective of the EXTRACT project is to explore innovative and comprehensive approaches that facilitate the development, deployment, and efficient execution of data mining workflows across a diverse, secure, and energy-efficient Compute Continuum. This initiative aims to address the unique challenges posed by extreme data characteristics. In this document, we will outline the overall requirements of the Continuum architecture that will be executed in one of the following environments: the Cloud, the Edge or the High-Performance Computing (HPC) systems.

2.1. Purpose and objectives

The objective of this document is to provide a description of the Compute Continuum architecture, encompassing Edge, Cloud and HPC components, which will serve as the foundation for the EXTRACT research.

Objectives:

1) Determine the selection of parallel and energy-efficient hardware architectures intended for utilization on the Edge and HPC sides.
2) Establish the EXTRACT Continuum framework, incorporating an interoperability abstraction layer while prioritizing data security in its management.

2.2. Relationship with other WPs

Deliverable	Task	Relation
D1.1	T1.1	Description of the use-cases and the requirements.
D2.1	T2.1	Requirements of the data infrastructure and data mining framework.
D3.1	T3.1	Data-driven orchestration requirements.

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, gives a main view of the structure of the document and describes the context giving a general overview of the Compute Continuum.
- Section 3 lists the requirements that influence the definition of both the hardware platform and the components of the Continuum architecture, including the security considerations.
- Section 4 details the integration process, incorporating the explanation of key aspects that will be followed throughout the project's execution.
- Section 5 provides an overview of the conclusions derived from this document.

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

2.4. Compute Continuum overview

In today's interconnected world, the Internet plays a vital role in people's lives. With the increasing accessibility to computers and smartphones, more and more people have become connected to the Internet. Simultaneously, devices have become capable of autonomously connecting to the Internet without human intervention, giving rise to the Internet of Things (IoT). This interconnectedness has led to the development of the Compute Continuum, which encompasses a range of computing devices and technologies.

The Compute Continuum is a concept that recognizes the diverse computing capabilities and resources available, spanning from Edge devices to Cloud infrastructure. It arises from the evolving needs of our interconnected world, where the hierarchical connection of objects has given rise to the IoT. As IoT devices have progressively gained computational power, the concept of the Compute Continuum has emerged to address the different computing requirements and challenges.

At one end of the Compute Continuum, we have Edge devices such as sensors, wearables and small computing devices. These Edge devices possess limited processing power and storage capacity but can perform basic computations locally. They are often found in scenarios where low latency and immediate response are crucial, such as real-time decision systems, patient monitoring or smart buildings.

On the other end of the Continuum, we have the Cloud infrastructure, which provides scalable and on-demand access to vast computing resources. Cloud Computing offers the ability to process and analyse large volumes of data, leveraging powerful data centres. It enables resource-intensive tasks and provides storage capabilities that are not feasible on Edge devices alone.

Between the Edge devices and the Cloud, various intermediate points exist along the Compute Continuum. These include gateways, Edge computing, Hybrid Cloud and High-Performance Computing. Gateways act as intermediaries between Edge devices and IoT devices, performing data aggregation and pre-processing tasks. Edge Computing extends the capabilities of Edge devices by utilizing intermediate devices like switches, routers and workstations to perform computations closer to the Edge. Hybrid Cloud integrates both public and private Cloud infrastructure, providing flexibility and control over sensitive data. HPC systems offer high computational performance for demanding workloads, while emerging technologies like Quantum Computing are also part of the Compute Continuum.

The Compute Continuum has evolved in response to several key needs.

1. **Efficient Data Processing**: The growing volume of highly heterogeneous data from diverse sources needs efficient data processing methods that can handle large amounts of data effectively and efficiently.

2. **Real-Time Decision-Making and Time-Sensitive Applications**: Applications such as IoT deployments and autonomous systems require real-time decision-making, capabilities to analyse and respond to data promptly. The Compute Continuum is particularly valuable for time-sensitive application that

require immediate action, enabling faster response times and reducing reliance on centralized Cloud processing.

3. **Reduced Latency and Bandwidth Constraints**: By leveraging Edge Computing within the Compute Continuum, organizations can minimize latency and bandwidth limitations by processing data closer to its source, resulting in improved response times, reduced network congestion and overall optimization of resources' usage.

4. **Scalability and Resource Utilization**: The Compute Continuum allows for fluid resource allocation and sharing across different platforms, enabling organizations to optimize the utilization of Edge devices, Cloud infrastructure and HPC systems, leading to improved scalability and operational efficiency.

5. **Interoperability and Integration**: Smooth data and computation flow between Edge, Cloud and HPC systems are essential. The Compute Continuum ensures interoperability and integration, facilitating seamless workflows and efficient collaboration.

6. **Security and Privacy**: Strong security measures are crucial to protect sensitive data throughout the computing ecosystem. The Compute Continuum emphasizes the implementation of security measures at each level to ensure cybersecure computations at all levels and sources, as well as data's privacy, integrity and confidentiality.

These needs drive the development and adoption of the Compute Continuum architecture, addressing the evolving requirements of modern computing environments.

3. Compute Continuum layer

3.1. Interoperability abstraction layer

The interoperability abstraction layer is designed from the early stages of the EXTRACT project to ensure the efficient exchange of data and services between Edge, Cloud and HPC. Therefore, the design needs to include principles of flexibility, scalability and adaptability to fit the needs of the Continuum.

It provides a standardised interface for deploying workflows, supports resource orchestration and dynamic provisioning, and integrates with various orchestration frameworks and resources. It is therefore necessary to establish a list of requirements for the nodes involved to match the desired level of abstraction.

It is important that these requirements are compatible with those from D3.1.

3.2. HPC requirements

High volume and speed characteristics are effectively addressed by the use of High-Performance Computing (HPC) technologies, which support massive parallel processing capabilities and advanced acceleration features (e.g., GPU, FPGA, many-

core fabrics, AI-cores), making HPC a key component of the compute continuum in EXTRACT. However, standalone HPC systems are not suitable for: (i) real-time operations and geographically disperse data sources, due to the potentially large communication latencies and single-point resource contention; (ii) energy-efficient solutions; (iii) extreme large-stored volumes. The dispersity of data enforces an interoperable system and data processing at the edge as well. The following are requirements applicable to HPC in the context of EXTRACT:

1) **Processing power**: the first obvious requirement for the HPC is that it should have sufficient processing power to handle the large datasets generated by the two use cases. Depending on the size and complexity of the datasets, this may require high-performance CPUs, GPUs, or other specialized hardware.

   While HPC systems are designed to provide high processing power, in EXTRACT it is important to focus on parallelization to ensure that computing resources are efficiently used. Parallel programming models, such as OpenMP [1] and CUDA [2], are well-suited for EXTRACTing performance from HPC systems, but they may not be optimized for data-intensive applications. To that end, we can leverage programming models and frameworks such as MapReduce, Apache SPARK or COMPSs, further described below.

   To enhance the parallelization process for data-intensive applications, it is necessary to incorporate data-awareness into the parallel programming models. This involves identifying when data is ready for execution and allocating resources accordingly using data flow analysis. By doing so, the parallel programming models can ensure optimal resource allocation and help to EXTRACT maximum performance from the HPC system.

   In addition to performance benefits, parallel programming models can also improve programmability, portability, and scalability. By hiding the complexity of the underlying platform, these models can ease the development of software for HPC systems and port software across different platforms. They can also be designed to scale effectively across multiple nodes, allowing for even greater performance gains on large-scale systems.

   One parallel programming model that can be used to enhance data-intensive applications on HPC systems is COMPSs [3]. COMPSs is a programming model that incorporates data-awareness into its parallelization process, allowing for optimized resource allocation and improved performance. Specifically, COMPSs uses a data-driven approach to parallelism, in which the data dependencies of an application are analysed to determine how to allocate resources and schedule tasks. This allows for the efficient use of the HPC resources and can help in avoiding resource contention and other performance issues.

   In addition to data-awareness, COMPSs is designed to be highly portable, allowing it to be used on a wide range of platforms. It is also highly scalable, allowing it to effectively utilize resources across multiple nodes. Finally, COMPSs provides a high degree of programmability since it supports multiple programming languages, making it easy to develop and maintain the software that uses it.

2) **Network bandwidth**: The HPC system should be connected to the edge and cloud components of the compute continuum with a high-bandwidth and low-latency data stream. Specifically, the data stream should be capable of fast transfer of large datasets and results between the different components. This is particularly important when dealing with data-intensive applications such as the collection of data in Venice to generate a personalized escape route in case of an emergency. To ensure efficient data transfer, the data should be pre-processed and filtered at the edge device to reduce the amount of data that needs to be transferred, so that large amounts of data do not cause a bottleneck if the network bandwidth is not sufficient.

   To address this, the HPC system should be connected to the edge and cloud components of the compute continuum with a high-bandwidth and low-latency data stream. This can be achieved with high-speed networking technologies, which are designed to provide high-bandwidth and low-latency interconnects between HPC nodes. In addition, the network should be designed to be fault-tolerant and resilient, to ensure that data transfer can continue even in the event of a network failure or any other issue.

3) **Security**: The HPC, as well the entire Compute Continuum architecture and its elements, must adhere to strict security requirements to ensure the confidentiality, integrity, and availability of the data and applications. This includes measures for example to prevent unauthorized access, ensure secure communication, protect against malicious attacks, and generally advised cybersecurity practices (e.g., ISO27k, OWASP, NIST). Such preventive measures are quite standard and well-studied in classical computing environments but require additional considerations in the context of HPC and Compute Continuum. Since the HPC will be processing sensitive data collected from edge devices, it is critical to establish proper authentication and authorization mechanisms to control access to the data, including at the source/edge, in transit, during HPC processing, and at rest. It is also important to implement proper (and thoroughly tested and validated) backup and disaster recovery mechanisms to ensure that the data and applications can be quickly restored in the event of a security breach or other unexpected event. Achieving these requirements may require specialized policies, practices (e.g., ISO27k, OWASP, NIST), software, protocols, cryptographic approaches, or technologies such as homomorphism, which allows for processing of encrypted data.

4) **Software**: The HPC should have the necessary software tools and frameworks installed to support the processing requirements of the two use cases, such as any required machine learning library, data modelling frameworks, etc. This can be achieved by leveraging container technologies such as Singularity [4], which enhances portability at the framework level. Moreover, "software security" is an integral part of the requirements and the architecture that includes considerations such as ensuring integrity of the software-supply chain, the integrity of the execution environments (e.g., docker containers), and overall software security practice (e.g., OWASP and similar).

3.3. Cloud requirements

Cloud computing is the crucial component of Compute Continuum platforms. Cloud exposes various services, unlimited resources and serverless platforms that became vital to address high workloads demands of applications during their execution phase. Hybrid clouds became a popular approach, where customers leverage private clouds installed on premise with public clouds that offer unlimited resources. As an example, edge frameworks can be installed on the edge and process initial raw data that once processed it moved to the public cloud for further processing, like Machine Learning workloads that usually requires many powers and GPUs.

To address the cloud requirements for EXTRACT Continuum platform, we need the following requirements:

1) A framework that can automatically provision a cluster of VMs (or connect to the existing cluster) in the cloud to address requirements of specific workloads, while using an optimization phase to decide on the VMs types and other configurations like memory, CPU or GPUs and decide on specific cloud and a region based on various strategies, like data locality. The framework should have a capability to fast provision additional VMs if needed in the runtime. This auto-scaling enablement is a crucial requirement to support workloads like cloud bursting, that requires massive compute power to support spikes in data processing.

2) A system capable to deploy a customer workload to the public or private cloud. System should be capable of leveraging multi cloud and multi region workloads, considering optimization strategies, like data locality, caching for frequently accessed data, perhaps network optimizations, policies, etc.

3) A storage system. Object storage is very popular platform for persisting Big Data and all Big Data engines can directly access data persisted in the object storage for read and write flows. Major public cloud providers are offering object storage with its unlimited scale, like Amazon S3 [5], IBM Cloud Object [6], and so on. Object storage can also be installed on premise in private data centres. In the EXTRACT Continuum we need both public object storage, like IBM Cloud Object Storage and private object storage, like CEPH [7] (with S3 API) that is installed on premise over K8s platforms.

4) A Container Orchestration Engine (COE) is a fundamental requirement for the EXTRACT platform to effectively support various workloads. A COE provides the necessary infrastructure for managing and orchestrating containers, enabling seamless deployment, scaling, and management of containerized applications. It allows for workload portability and flexibility by abstracting the underlying infrastructure and providing a consistent interface for deploying applications across different container platforms. The COE ensures that applications can be agnostic to specific technologies such as Kubernetes (K8s), enabling them to be deployed in a variety of environments, including public cloud, on-premise, or edge deployments.

5) A single access point for multi cluster deployments, covering hybrid clouds, like K8s deployed on premise and public cloud services.

6) Monitoring is a crucial component in almost every architecture. An ability to monitor executions, provisions, component and failures is what makes any platform to efficiently handle Big Data workloads. EXTRACT Continuum platform should be connected to monitoring capabilities and is required to efficiently handle partner use cases of EXTRACT.

Based on the requirements above we decided to leverage Sky framework to be part of EXTRACT Continuum Platform. Sky computing is the recent trend that comes to address multi cloud workloads to achieve cost efficiency, better resource utilizations, etc. Sky is not just multi-cloud, where application is aware of multiple cloud. Sky is a new concept where application is cloud agnostic and Sky framework decides on its own which clouds and regions to deploy and execute this application while optimizing costs, latency, data locality, etc. In particular, SkyPilot [9] is a novel framework for easily and cost effectively running ML workloads on any cloud. SkyPilot abstracts away the cloud infra burden: launch jobs & clusters on any cloud (AWS, Azure, GCP, Lambda Cloud, IBM, Samsung), find scarce resources across zones/regions/clouds, Queue jobs & use cloud object stores.

Requirements above are well connected to EXTRACT use cases and other partner's requirements. As example requirement on object storage is crucial for TASKA use-case as they already employ CEPH with S3, and object storage is a designated feature of the data staging layer. The requirement on COE aligns well with the orchestration layer (D3.1) that assumes K8s as the infrastructure, and the requirement on monitoring is well connected to D3.1 as well.

3.4. Edge requirements

These requirements specifications outline the key functional and non-functional requirements for EXTRACT focused on edge device and user application management. The functional requirements emphasize the need for seamless device provisioning, centralized monitoring and control, application lifecycle management, secure over-the-air updates, and advanced analytics and insights. These requirements ensure efficient onboarding, real-time visibility, end-to-end application management, secure updates, and data-driven decision-making capabilities. In addition, the non-functional requirements address the runtime aspects of the edge devices and user applications. Performance and scalability requirements ensure fast execution of user applications and the ability to handle increasing workloads. Reliability and availability considerations focus on minimizing downtime and ensuring continuous operation, even in the face of device failures or network disruptions. Resource efficiency requirements emphasize optimal resource utilization, and platform compatibility requirements ensure seamless integration across heterogeneous environments.

The below requirements are based on the premise that users package their applications as Docker/OCI containers to run under Container Orchestration Engine (COE) on the edge devices. The following outlines the elements of the edge management system that provides the runtime and management capabilities for execution of the user-defined applications at the edge:

- SaaS/PaaS for edge device and application management (Edge SaaS)
- Edge device and application management Agent (Edge Agent)
- Container Orchestration Engine (COE)

- Container Runtime (CR)
- Operation System (OS)
- Edge device hardware (edge HW)

3.4.1. Functional requirements

1. **Seamless Device Provisioning**: The provided solution should support automated and simplified onboarding of edge devices into the system. This includes streamlined device registration, authentication, and configuration processes to ensure quick and hassle-free integration with the edge computing infrastructure.

2. **Centralized Monitoring and Control**: The provided solution should enable centralized monitoring and control of edge devices and user applications. It should provide real-time visibility into the status, health, and performance of connected edge devices, allowing operators and application developers to proactively identify and resolve issues.

3. **Application Lifecycle Management**: The provided solution should support end-to-end management of user applications deployed on edge devices. This includes features such as application deployment, version control, scaling, and updates.

4. **Secure Over-the-Air Updates of Edge Devices**: The solution should provide secure mechanisms for performing over-the-air updates to the edge device. This ensures that devices are running the latest software versions with patches and bug fixes, while minimizing downtime and operational disruptions. The edge management solution must provide (elements of) edge device management functionalities, like edge device halt, reboot, OS, COE, and CR upgrade and configuration.

5. **Rich web-GUI**: The provided solution must provide edge operators, application developers and users with rich web-GUI for edge device and application management.

6. **Telemetry collection**: The system should provide collection of the edge device and applications telemetry to the centralized storage. The collection of the telemetry must be done in a PUSH mode from the edge devices to the storage. A secure communication channel is assumed.

7. **Analytics and Insights**: The solution should offer analytics and insights for edge devices and user applications. It should provide data-driven visibility into device performance, resource utilization, and application behavior.

8. **COE-enhanced Edge Devices**: The system should provide edge devices with COE running on them.

9. **Support for Various COE at the Edge**: Edge application management functionality of the provided system must be extendable to support various COE implementations.

10. **Edge device OS supporting selected Container Runtime**: Selected OS on the edge device must support Container Runtime of the selected COE.

11. **Remote orchestration and management of fleets of independent edge devices**: The system must allow for remote orchestration and management of fleets of independent edge devices and provisioning and lifecycle managing user applications. This assumes the ability of the system to hold the inventory of the edge devices not clustered on COE level neither among themselves nor with the system management platform.

12. **Remote orchestration and management of the COE clusters at the edge**: The system must allow for remote orchestration and management of COE clusters at the edge and deployment and lifecycle management of user application workloads on them.

13. **Cross-edge device application service discovery**: There must be a possibility for service discovery between user applications running on different edge devices under the supervision of the different not clustered COEs.

14. **Edge device and workload management on edge devices behind NAT**. Edge and application workload management system must provide complete edge and application manageability on the edge devices running behind NAT. This can either be achieved by running edge management agents in PULL mode or by utilizing a managed or public VPN service (with corresponding VPN client at the edge).

15. **Application image pre-fetch in case of restricted network conditions**: The edge application management system must be able to perform the user application deployment and upgrade in two stages: image pre-fetch and then application rollout. In case of low throughput and/or intermittent network availability, the edge application management must be able to pre-fetch (with image download resume functionality) all the images user defined as part of their application deployment definition and only then proceed with the application deployment or upgrade.

16. **Peripheral discovery**. The Edge management system must be able to:
    a. identify the peripherals connected to the edge device.
    b. configure the edge device so as the edge applications can seamlessly access the peripherals connected to the edge device.

3.4.2. Non-functional requirements

For the below set of the non-functional requirements the actual KPIs following the concrete runtime requirements from the project's Use Cases will be collected at a later stage of the project. The selected set to of tools that constitute the edge computing solution must be deployed and configured to fulfil the KPIs.

1. **Footprint of the COE and CR at the Edge.**
   In case of multiple choice, the selected combination of COE and CR must have minimal footprint.
2. **Performance and Scalability.**

The edge computing solution should demonstrate high performance, with minimal latency and response times, to ensure smooth and efficient execution of user applications.

3. **Scalable Edge Management System.**

   The Edge management system must be scalable to accommodate a growing number of edge devices and user applications, without compromising performance or stability.

   The system's infrastructure should be capable of handling increased workloads and scaling resources dynamically to meet varying demands.

4. **Reliability and Availability.**

   - The solution should have high reliability, ensuring that edge devices and user applications are consistently available and operational.
   - It should include mechanisms to handle device failures, network disruptions, and other potential issues, minimizing downtime and ensuring continuous operation.

5. **Efficient Use of Resources.**

   - The solution should optimize resource usage, including CPU, memory, and network bandwidth, to maximize the efficiency of edge devices and minimize operational costs.
   - The solution should employ local intelligent resource management techniques, such as load balancing and resource allocation algorithms, to ensure efficient utilization of available resources.

3.5. Security requirements

Below we outline the general high-level security requirements, split by their domain of direct application to various parts and elements of the EXTRACT data, software and infra-architecture.

3.5.1. Cybersecurity requirements

Software integrity

Ensure/enforce external/third-party supply-chain software security and SBOM practices are followed:

1) Strong-check integrity and authenticity of ALL downloadable/cached packages, libraries, source-code-to-be-compiled, pre-compiled-binary-code, etc.
2) Strong-check integrity and authenticity of ALL package sources (e.g., PyPI, NPM, APT, etc.)
3) Continuous DevSecOps scan of external/third-party supply-chain for malicious or threat embedded elements

Ensure/enforce EXTRACT own packages, modules, tools, source and binary codes follow and implement "self-software-integrity" provisions and checks.

Secure deployments
Ensure/enforce the deployments of each computational unit is performed in secure manner.

Secure computations
Apply generically applicable principles from: Secure Deployments.

Resilience to software/computation attacks
Explore feasibility and possibilities to include elements (mainly software) that can enable various modules of EXTRACT architecture to withstand software and computational attacks (e.g., denial of service, slowloris, buffer overflows, code injections, untrusted code execution, etc.).

Security of edge/sensors/nodes
1) Apply DevSecOps CI/CD cybersecurity scanning of nodes, including blackbox, whitebox, graybox, pentesting and fuzzing.
2) Apply source-code/binary-code SBOM mapping and analysis of each node/computing-environment.
3) Apply generic/classical threat and network scanning to continuously map, assess and understand potential attack surface coming from the known/running configurations of each edge/sensors/node's type/instance.

Software code security quality checks
1) Periodic human code reviews (especially critical code)
2) Automated code reviews using static-analysis tools (e.g., Veracode, Checkmarx, etc.)

Other considerations
Apply also considerations from T4.3 Cybersecurity.

3.5.2. Data security requirements

Data confidentiality
Data-in-transit:

- Ensure/check/enforce TLS 1.2+ is used on all exposed endpoints (e.g., API, web interfaces, etc.)
- Ensure/check/enforce proper certificates and certification chains are used on each exposed endpoint.
- Ensure/check/enforce only strong crypto ciphers are used (e.g., TLS_AES_128_GCM_SHA256) and no weak or downgrade ciphers are allowed.
- Apply/perform/enforce automated/periodic scans for usage of weak/insecure/non-compliant crypto ciphers and crypto material (e.g., certificates, certificate chains).

Data-at-rest:

- Apply/follow/enforce best-practices such as [10].
- Apply/follow best practices such as NIST "Protection of Data at Rest" series (e.g., [11])

Data-in-processing:

- Apply generically applicable principles from: Data-in-transit and Data-at-rest, Secure Computations
- Apply applicable principles from:
  - Software integrity
  - Secure deployments
  - Secure computations
  - Resilience to software/computation attacks

Data integrity

- Apply/follow/enforce best-practices such as [10] and [12].
- Ban/deprecate/fail use of weak integrity check functions (e.g., MD5, SHA1)
- Apply/perform/enforce automated/periodic scans for usage of weak/insecure/non-compliant function use.

Secrets management

- Apply/follow/enforce best-practices such as [13], both in manual review as well as part of automated DevSecOps tooling
- Apply/perform/enforce automated/periodic scans for "weaks and leaks".

Other considerations

Apply considerations from task T2.4 Data Security

3.5.3. Security of ML models

Mainly apply considerations from task **T2.4 Data Security**

3.6. Compute Continuum layer high level design

As stated in the Compute Continuum overview, its aim is to group together heterogeneous devices with diverse capabilities such that each task of a workflow is executed on the best device depending on its requirements. E.g., real-time ingestion of data in edge devices or a highly parallelizable algorithm in HPC. Kubernetes is one of the selected technologies as part the solution to implement the Compute Continuum layer for the EXTRACT project, especially in the Cloud and HPC.

Kubernetes is the most widely used solution for container orchestration in cloud-based environments. It provides a highly scalable and flexible infrastructure for deploying containerized applications, as well as automatic load balancing and service discovery, ensuring traffic is efficiently distributed across containers and services.

However, although it is being increasingly adopted for edge and HPC, there are still some challenges that the EXTRACT platform will have to solve. As for HPC, the nodes should run Singularity as container runtime engine [4], instead of container or Docker, which are the ones typically used in cloud.

As for the edge, Kubernetes is resource-intensive and requires significant computational resources to operate and it is not optimized for this kind of environments with high latency and low bandwidth connections. So, although it is designed to be highly scalable, it may not be the best option for edge environments

that require massive scale-out capabilities due to its resource requirements and complexity.

The following are some container orchestration solutions and edge management middlewares that have been considered for the edge devices:

- **KubeFed** [14]: It allows the configuration of multiple Kubernetes clusters from a single set of APIs in a hosting cluster. However, we discarded KubeFed because it is currently an archived project.

- **K3s** [15]: K3s is a lightweight distribution of Kubernetes that is designed for edge computing and IoT (Internet of Things) use cases. It is designed to be easy to install and operate on resource-constrained environments, and it provides many of the core features of Kubernetes.

- **OpenShift and Rancher** [16] and [17]: among other reasons, both OpenShift and Rancher are discarded as they are both commercial offerings that require licensing and ongoing support costs.

- **Kubeflow** [18]: Kubeflow is a Kubernetes-based platform especially designed for building and deploying machine learning workflows. It provides a set of tools and libraries for building and deploying ML models. Kubeflow can be used in a wide range of environments, including edge computing and HPC. The drawbacks of Kubeflow in the context of EXTRACT, are that first, not all applications in the edge are based in machine learning for which Kubeflow is optimized. Furthermore, Kubeflow does not scale and has a high resource usage as well.

- **MicroK8s** [19]: MicroK8s is a lightweight, fast, and secure distribution of Kubernetes that is designed for development, testing, and edge computing use cases. It can be installed and operated on resource-constrained environments, and it provides many of the core features of Kubernetes. It allows running a complete Kubernetes environment on a single node, with features such as DNS, ingress, and dashboard. MicroK8s is also optimized for resource-constrained environments. However, microK8s is typically used for building and testing Kubernetes-based applications in a local development environment, rather than for deploying and managing applications at scale in a distributed infrastructure such as in the use cases of EXTRACT.

- **KubeEdge** [20]: KubeEdge is specifically designed for edge computing scenarios and has been developed with features that address the unique requirements of edge environments. It extends Kubernetes by allowing nodes to be deployed on edge devices and providing a way to manage and orchestrate those nodes from a central location. KubeEdge uses the QUIC protocol for communication among clusters, instead of TCP such as Kubernetes or K3s. This is enhanced for a cloud/edge communication architecture. It also enables edge devices to act as Kubernetes nodes and provides the ability to deploy containerized applications and services to those nodes. This allows edge computing workloads to be managed and orchestrated in the same way as cloud-based workloads, using the same tools and APIs. KubeEdge also provides additional features such as device management, edge data analytics, and machine learning capabilities. Finally, it integrates well with other Kubernetes-

based platforms too, providing a seamless experience across multiple environments.

- **Nuvla/NuvlaEdge** [21]: Nula/NuvlaEdge is a multi-tenant edge-to-cloud solution composed of two components: Nuvla and NulvaEdge. *Nuvla* is an edge management software. It's an open-source solution and is available as either a stand-alone software stack for installation on premises or as a managed PaaS via *Nuvla.io*. Nuvla enables users to manage their edge devices and deploy containerised applications at the edge and cloud. Independent Edge devices and Cloud Computing instances which are equipped with Kubernetes or Docker COE can be onboarded and used to provision containerised applications. Nuvla features applications marketplace and meta-data catalogue for describing user data located at the edge and cloud. Nuvla provides users with the data discovery capabilities and staging applications where the data is located. Nuvla features rich user-friendly web GUI and a comprehensive RESTful API for third-party service integration.

  NuvlaEdge is an Edge framework runtime software, which transforms any device into remotely managed Edge device. This allows the user (through Nuvla) to connect to and monitor each edge device individually. NuvlaEdge provides facilities for: VPN-based networking with Nuvla and users (typically operators of edge device), MQTT-based internal messaging, application monitoring, security and discovery of attached HW components, such as network devices, GPU boards, etc. All edge and application management operations are available in PUSH and PULL modes (allowing for network restricted use cases). It supports Kubernetes and Docker (Swarm) COE.

- **OKD** [22]: OKD is an open-source sibling Kubernetes distribution to Red Hat OpenShift. It is a container application platform that provides a complete solution for deploying and managing applications in a compute continuum environment. OKD is based on the Kubernetes container orchestration engine and extends its capabilities with additional features and tools.

  OKD offers a flexible and scalable platform for deploying containerized applications across various deployment environments, including public and private clouds, on-premises data centers, and edge devices. It provides a unified and consistent experience for managing applications throughout their lifecycle, from development to deployment and scaling.

  One of the key features of OKD is its focus on developer productivity. It provides built-in support for continuous integration and continuous delivery (CI/CD) workflows, allowing developers to easily build, test, and deploy applications using automated pipelines. OKD also offers a rich set of developer tools and integrations, making it easier to develop, debug, and monitor applications in a compute continuum.

  In addition, OKD provides robust security and multi-tenancy capabilities, allowing to enforce access controls, isolate applications, and ensure data privacy within their compute continuum environment. It offers built-in authentication, authorization, and encryption mechanisms to protect sensitive data and resources.

In this respect, our choice was limited to the three tools: KubeEdge, OKD and Nuvla/NuvlaEdge.

One of the key benefits of these options is their ability to operate in offline or intermittent connectivity scenarios, which are common in edge computing environments. These three tools provide local storage and synchronization capabilities that allow edge nodes to continue operating even when they are disconnected from the central manager. KubeEdge provides a distributed message bus that enables communication between edge nodes without relying on a centralized messaging service, reducing latency and improving resilience. All of them support Kubernetes.

However, unlike Nuvla/NuvlaEdge, KubeEdge does not support functional requirements 4, 5, 6, 7 and 16 listed in section 3.4.1.

A set of field deployments and evaluations of KubeEdge, OKD and Nuvla/NuvlaEdge is planned to be conducted in the forthcoming stage of the project to select the most suitable solution or a combination of them. The experiments will be taking into account the edge and application management capabilities, data discovery and the applications staging to the data requirements defined in D2.1 and the usability aspects.

Kubernetes update configuration mode

In Kubernetes, as in other orchestrating technologies, the scheduling of workloads can be handled in different ways, namely push, pull, or hybrid modes.

1) In pull mode, the Kubernetes worker nodes proactively ask (pull) the master for new workloads, rather than waiting for the master to push workloads to them. This can be useful in scenarios where network connectivity is limited, as the worker nodes do not need to maintain a constant connection to the master asking for new jobs.
2) In push mode, the Kubernetes master pushes workloads to the worker nodes as they become available.
3) Hybrid mode is a combination of push and pull modes, where the worker nodes periodically check in with the master for new workloads, while also receiving workloads pushed by the master.

Each model has its strengths and drawbacks, making it the best alternative under certain circumstances. The following table summarizes the main pros and cons of each model:

Model	Advantages	Disadvantages
Push Mode	It can be faster and more responsive as the master node can immediately send tasks to the workers as they become available, without waiting for the workers to request them. Push mode can reduce network traffic, as the workers do not need to constantly poll the master node for new tasks.	It may not be suitable for very large-scale deployments with a huge number of workers, as the master node can become overwhelmed trying to push tasks to all the workers. Push mode may require a more complex implementation, as the master node needs to keep track of which workers are available and

		which tasks they are working on, in order to push tasks efficiently. It may not be as fault tolerant as pull mode, as there is a risk of tasks being lost if a worker node fails before it has completed its task.
Pull Mode	Worker nodes only request tasks when they are connected and "free", which can optimize resource usage and reduce unnecessary overhead. Can handle worker node failures more gracefully, as failed nodes will stop requesting tasks and can be replaced by new nodes.	Task scheduling can be delayed if worker nodes do not request tasks frequently enough or if there is a backlog of pending tasks. Not well-suited for real-time or latency-sensitive workloads, as tasks may not be executed immediately. Requires additional configuration to ensure that worker nodes are properly scaled up and down to handle changes in demand.
Hybrid Mode	Combines the benefits of push and pull modes, allowing for more flexibility in job distribution. The master node can push urgent or time-sensitive tasks to worker nodes, while allowing them to pull other tasks when they have the capacity. Can help avoid under-utilization of worker nodes by ensuring that they always have tasks available to work on. Reduces the need for constant communication between the master and worker nodes, which can help to reduce network traffic and improve overall system performance.	Can be more complex to set up and manage than push or pull mode alone. Requires careful coordination and planning to ensure that tasks are distributed efficiently and that there is no duplication or overlap. May require additional monitoring and troubleshooting to ensure that all nodes are working effectively and that tasks are being executed correctly. May not be the best option for all types of applications and may require more customization and tweaking to work optimally in some cases.

Table 2. Comparison between models

As depicted above, choosing the hybrid mode for deployment in the compute continuum combines the best of both worlds, striking a good balance of the benefits of push and pull modes while minimizing their drawbacks. In hybrid mode, critical tasks can be pushed to the workers, while the rest of the tasks can be pulled as needed, reducing the overhead in the system, which is a suitable choice for use cases with varying task requirements and resource utilization. This mode allows for greater flexibility and control over the deployment of applications, as well as more efficient utilization of resources. By choosing hybrid mode, EXTRACT can ensure that critical

tasks are deployed efficiently and with minimal latency, while at the same time maintaining a scalable and efficient deployment process for non-critical tasks.

When operating in pull mode (or hybrid for that matter), it is essential to have a sufficiently high frequency at which the nodes poll for workload. The frequency of polling depends on several factors, including the expected workload and the available resources. If the frequency is too low, the nodes may not receive new tasks in a timely manner, which can lead to decreased efficiency and utilization of resources, especially if dealing with critical or real-time tasks. On the other hand, if the frequency is too high, it can lead to unnecessary network traffic and increased overhead, which can also affect performance. Therefore, it is important to strike a balance between the frequency of polling and the workload requirements to ensure optimal performance and utilization of resources. This can be achieved and tweaked by monitoring the system's performance and adjusting the polling frequency accordingly.

Compute Continuum layer topology

Topology	Description	Disadvantages
Single-cluster	Only one cluster with multiple masters where all three types of nodes: HPC, Cloud and Edge are connected to the same control plane.	Can be a problem with long-distanced nodes (K8s does not work well outside LAN).
HPC-Cloud multi-cluster	Multi-cluster with masters (control planes) in HPC and Cloud and edge nodes connect to the Cloud. Must create overlay network among clusters.	All edge nodes would connect to the Cloud. Good scenario to test EdgeMesh.
HPC-Cloud-Edge multi-cluster	Multi-cluster with masters (control planes) in HPC and Cloud and Edge. Nodes connect to the nearest master. Must create overlay network among clusters.	Nodes deployed in the same geographical area could form a cluster.

Table 3. Compute Continuum layer topology

Compute Continuum layer overlay network

Kubernetes ensures that all pods on a single cluster can communicate with each other, independently on the node they are running on. But when dealing with multi-cluster architectures, an overlay network must be deployed by the administrators, so pods among different clusters can also communicate.

The following table is a sum up of the technologies that could be a good fit for the Compute Continuum layer of the EXTRACT project.

Topology	Description	Disadvantages
Edgemesh	Edge service mesh that focuses on providing service discovery and routing capabilities for applications deployed on edge devices.	Recommended by KubeEdge.
Istio	Full-featured service mesh that provides service discovery, traffic management, security, and observability for applications, among others	Most popular solution in Kubernetes.
Skupper	Service mesh designed to work across multiple clusters and cloud providers.	

Table 4. Compute Continuum layer overlay network

4. Initial Integration Plan

The EXTRACT project encompasses a distributed team consisting of multiple individuals from diverse institutions and fields of expertise. Given the varied nature of the project's tasks, conventional frameworks like Scrum or Kanban will not be strictly adhered to or imposed at the project level. However, each partner has the flexibility to organize their tasks by adopting these or other suitable frameworks.

Instead, the development of the EXTRACT product will be divided into distinct phases, wherein each phase will involve the integration of components from different work packages. It is highly advisable for partners to start working on component integration as early as possible, employing illustrative examples that facilitate understanding of the interfaces between the various technologies employed in each work package. Subsequently, the complexity of the examples or datasets can be incrementally increased until the testing of actual use cases becomes feasible.

The subsequent section is structured as follows: Section 4.1 presents the development and integration processes, as well as quality assurance measures for the EXTRACT project; Section 4.2 outlines the designated infrastructure intended to facilitate information sharing and coordination among all teams; and finally, Section 4.3 details the established standards and guidelines designed to streamline the utilization of the provided infrastructure.

4.1. Processes

This section introduces two primary processes: (1) the development and integration process, and (2) the quality assurance process. The development and integration process aims to establish a continuous development approach that mitigates risks and streamlines the building and releasing procedures. On the other hand, the quality assurance process focuses on ensuring the delivery of high-quality development outcomes. The subsequent subsections provide detailed specifications of each process, including the corresponding activities associated with them.

4.1.1. Development and Integration Process

The EXTRACT project encompasses various components that constitute the EXTRACT software ecosystem. These components collaborate through predefined interfaces to deliver diverse functionalities. The software architecture integration process involves combining all software components into a unified ecosystem, ensuring that each component operates according to the specified functional requirements. This integration process aims to establish a cohesive software architecture that offers the desired functionalities.

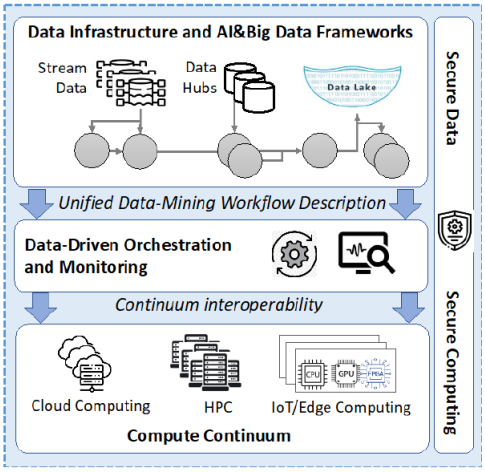

Figure 1. Main EXTRACT platform components

The development and integration process in the EXTRACT project is divided into four distinct phases. Figure 2 illustrates the integration process of the different components within the EXTRACT ecosystem, considering the various phases and all components. The integration follows a tree structure and is performed incrementally, gradually incorporating each component into the overall ecosystem.

Phase 1 of the EXTRACT project spans the first six months and focuses on establishing a solid foundation for the project. During this phase, the project team defines the complete set of functional and non-functional requirements. Additionally, the architecture between the components is designed, and a preliminary selection of the technology stack is made. This phase also includes establishing effective communication channels between the different components.

As Phase 1 nears completion, efforts are directed towards the development of all deliverables, including the current document. These deliverables are scheduled to be finalized by Milestone 1 of the project, marking an important milestone in the overall project timeline.

As Figure 2 depicts, Phase 2 of the EXTRACT project focuses on the development of the different components based on the requirements defined in Phase 1. Phase 1 simply constitutes the definition of the requirements. On Phase 2, each Work Package is treated separately, where the output of one component serves as the input for the next component, with data or processes handled accordingly.

The Work Packages in Phase 2 include the Data Analytics and Data Mining stage, the

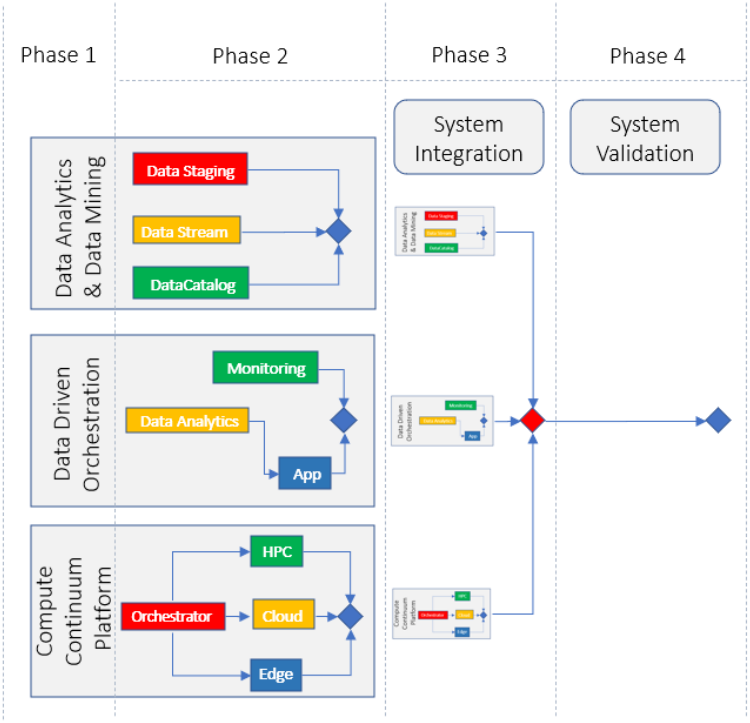

Figure 2. Integration phases

Data-Driven Orchestration, which encompasses analytics, and the Compute Continuum Platform. Additionally, monitoring across the entire Continuum is addressed.

During Phase 2, the functionalities of each component and the integrated components are verified through the use of unit tests. By the end of this phase (Milestone 2), each team is expected to meet the functional requirements specific to their corresponding component.

In Phase 3 of the EXTRACT project, a similar integration process as in Phase 2 will be repeated, but this time considering the different Work Packages as individual components. This integration effort will require close coordination among all teams to ensure the successful integration of all components using either a sample dataset or an initial real use case example.

For each integration activity, a designated leader will be responsible for overseeing the integration process. The integration will be validated through a combination of unit tests and, if necessary, regression tests to ensure the integrity and functionality of the integrated components.

During the 4th and final phase of the EXTRACT project, the validation of all software components within the EXTRACT software platform will take place. The micro-

validations conducted during the integration phases will help minimize unexpected issues at this stage. The primary objective of this phase is to validate and verify the entire platform, ensuring its functionality and compliance with the specified use cases.

All reported bugs and issues from previous phases will be addressed and resolved by the end of this phase, resulting in a project that is free from known errors. This comprehensive validation process encompasses both the individual components and the system as a whole, ensuring the fulfilment of functional and non-functional requirements.

4.1.2. Quality Assurance Process

The EXTRACT project emphasizes high-quality standards and places importance on effective communication and interconnection among its distributed teams. To ensure fault-free software and adherence to specified behaviour, several activities will be conducted throughout the development and integration process. These activities will occur at different stages and can be categorized as continuous or recurring. The following section provides an overview of the quality assurance activities performed within the EXTRACT project's development and integration process.

4.1.3. Unit Testing

During Phase 1, sample datasets provided by the use case leaders will be utilized for unit testing purposes. These datasets, or representative subsets of them, will be used in conjunction with unit tests to validate the implemented functionalities of each component and the interconnections between adjacent components. Unit tests play a crucial role in determining the correctness of individual code units, streamlining the integration process, and facilitating future modifications.

To maintain the quality of unit tests, it is important to adhere to certain guidelines. These guidelines include appropriately naming tests, keeping tests small and fast, covering boundary cases, and preparing tests for code failures. Initially, unit tests will be manually tested by the respective developers of each component. Depending on the programming language used for a particular component, a suitable testing framework such as JUnit (for Java) or GoogleTest (for C/C++) can be employed.

In the second stage, to ensure the effectiveness of unit testing, the tests should be automated using a continuous integration system, as explained in the upcoming subsections. This automation will enable the execution of the entire unit test suite upon code check-in, ensuring that both new and existing tests run successfully.

4.1.4. Regression Tests

During the integration process, it may be necessary to make changes to the implementation. In such cases, regression tests play a crucial role in ensuring that the modifications do not introduce any issues or break previously functioning components. Regression tests involve re-running previous tests each time a modification is made, as well as writing new tests when necessary. It is essential to maintain adequate test coverage to effectively conduct regression testing.

To conduct effective regression testing, several strategies and good practices should be followed. These include checking for possible side effects when fixing bugs, writing

regression tests for each bug that is fixed, and removing redundant tests. By following these practices, the regression testing process can provide confidence that the modifications do not negatively impact the existing functionality of the software ecosystem.

4.1.5. Bug and Issue Tracking

Bug and issue trackers are essential tools for managing and resolving software bugs and issues. These applications facilitate the creation, updating, and resolution of project issues, providing a platform for maintaining a knowledge base that contains information to aid in issue resolution. During the integration phase of the EXTRACT project, effective bug and issue tracking is crucial for ensuring smooth collaboration and communication among the different teams involved in integrating each component.

Using an issue tracker offers several advantages over real-time messaging or email-based communication. It provides a centralized and structured approach to managing tasks and issues, ensuring that important details are documented and easily accessible. Issue trackers enable teams to track the status of bugs and issues, assign responsibilities, and prioritize tasks effectively.

For the EXTRACT project, GitLab will be utilized as the chosen bug and issue tracking platform. GitLab offers a comprehensive set of features that support efficient issue management, enabling teams to effectively track, address, and resolve software-related problems. By utilizing GitLab as the issue tracking tool, the project aims to enhance collaboration, streamline issue resolution processes, and maintain a comprehensive record of project-related tasks and discussions.

4.2. Infrastructure

This section outlines the tools and platforms identified for potential use within the EXTRACT project. Infrastructure decisions are made based on consortium agreements and align with common development standards, as well as the quality assurance and integration processes described in this section.

4.2.1. Development Platform:

The development platform plays a crucial role in enabling efficient and collaborative software development within the EXTRACT project. The following platforms have been identified for use in the project:

Version Control System (VCS): The project will utilize Git as the version control system for managing source code. Git provides a distributed and scalable platform for code versioning, facilitating collaborative development and code management across distributed teams. GitLab, a web-based Git repository manager, will be used to host the Git repositories and provide additional collaboration features such as issue tracking, merge requests, and code review.

Integrated Development Environment (IDE): Each development team within the project is free to choose their preferred IDE based on their specific requirements and expertise. Commonly used IDEs such as IntelliJ IDEA, Eclipse, Visual Studio Code, and

PyCharm are recommended for their extensive features and compatibility with multiple programming languages.

Collaboration and Communication Tools: To foster effective collaboration and communication among team members, the project will leverage tools such as Slack, Microsoft Teams, or similar platforms. These tools provide real-time messaging, file sharing, and collaboration features that facilitate seamless communication and coordination among distributed teams.

Build and Continuous Integration (CI) Tools: The project will utilize build and CI tools to automate the building, testing, and integration of software components. Popular tools like Jenkins, Travis CI, or GitLab CI/CD pipelines can be adopted to set up automated build and integration workflows, ensuring the stability and reliability of the software ecosystem.

The selection and configuration of the development platforms will be performed by each team, considering their specific needs and preferences while adhering to the overall project guidelines and standards.

4.2.2. Instant Messaging and Transparency

Effective communication and transparency are crucial in the development and integration process, especially when working with remote teams comprising multiple individuals. To facilitate seamless communication and promote transparency, the EXTRACT project will utilize Slack as a team communication tool. Slack offers various benefits that contribute to efficient collaboration and information sharing:

Centralized Communication: Slack integrates all team communications into a single platform. It allows for the creation of channels, organized by topics, where different users can be assigned based on the visibility required for each channel.

Integration with Web Services: Slack seamlessly integrates with other web services such as GitHub, enabling notifications and easy access to view code check-ins. Additionally, it supports integration with file sharing services like Dropbox and Google Drive, facilitating efficient sharing of project-related files.

Comprehensive Search Functionality: Slack provides a powerful search feature that allows users to search for specific content across all channels and conversations. This functionality makes it easier to locate and retrieve relevant information, even in situations where large volumes of communication have taken place.

Code Snippet Sharing: Slack offers the capability to share code snippets with syntax highlighting. This feature enables team members to share code snippets for review, feedback, or collaborative editing. Other members can download the code snippet, view it in raw mode, or provide comments and modifications.

4.2.3. Integration Platform

This section defines the platforms that will be used for the integration of the EXTRACT project.

Automated Build System

The integration of the EXTRACT project will involve the use of an automated build system to streamline the build process and ensure consistency across different

components. While the specific technology for the automated build system has not been determined, one example is Apache Maven. Apache Maven is a project management and comprehension tool that centralizes the building, reporting, and documentation processes.

The primary objectives of implementing an automated build system within the EXTRACT project are as follows:

- **Ease of Build Process**: The automated build system aims to simplify the build process by providing predefined configurations and workflows, reducing the manual effort required for building and packaging the project components.

- **Uniformity**: By adopting an automated build system, the EXTRACT project aims to establish a standardized build system that ensures consistency across different components. This helps in managing dependencies, version control, and maintaining a cohesive project structure.

- **Quality Project Information**: The automated build system generates comprehensive reports and project documentation, providing valuable insights into the project's status, code quality, and potential issues. This information assists in identifying areas for improvement and ensuring project quality.

- **Best Practices Development**: The automated build system incorporates guidelines and best practices for development. It enforces coding standards, performs static code analysis, and facilitates the implementation of quality assurance measures, thereby promoting high-quality software development.

- **Smooth Migration to New Features**: The automated build system is designed to support seamless migration to new features and technologies. It provides the necessary tools and frameworks to handle changes and updates in dependencies, libraries, and configurations, ensuring the project remains up to date.

Continuous Integration System

To facilitate the integration and testing of software components in the EXTRACT project, a continuous integration system will be implemented. While Jenkins is one potential option, the specific technology has not been finalized yet. The chosen continuous integration system will have the following key objectives:

- **Continuous Building and Testing**: The system will automate the building and testing of software components on a regular basis. This ensures that any changes made to the project are quickly integrated and tested, allowing for early identification of integration issues and bugs. The automated process enhances productivity and reduces manual effort.

- **Monitoring External Job Executions**: The system will provide monitoring capabilities for externally-run jobs, such as cron jobs or jobs executed on remote machines. It will collect and store the results of these jobs, enabling developers to review them and take necessary actions. Notifications, such as email alerts, may be utilized to keep developers informed about job statuses and outcomes.

4.2.4. Quality Assurance Tools

This section covers the quality assurance tools used within the EXTRACT project to provide means to test and control code and thus system quality.

Issue Tracking Tool

The chosen tool for issue tracking in the EXTRACT project is GitLab. GitLab provides a comprehensive platform for managing and tracking issues and feature requests. It offers the following capabilities:

- **Issue Management**: GitLab allows for effective collaboration and definition of specific business needs. It facilitates tracking effort, size, complexity, and priority of issue resolution. By using GitLab, silos can be eliminated, and cross-functional engagement can be enabled.

- **Visualizing Work with Issue Boards**: GitLab provides issue boards that visualize the status of work across the entire lifecycle. It enables efficient management, assignment, and tracking of workflow. Agile delivery methodologies such as Kanban and Scrum can be effectively supported.

- **Maintaining Traceability through the DevOps Pipeline**: GitLab allows for linking issues with the actual code changes needed to resolve them. It provides visualization and tracking of the status of builds, testing, security scans, and delivery. This helps in maintaining traceability and enables the entire team to have a common understanding of the project's status.

4.2.5. File Storage and Collaboration

Within the EXTRACT project, we will utilize B2DROP from BSC as our file storage and online document collaboration platform. B2DROP offers secure and reliable cloud-based storage for research files, along with features for seamless collaboration among project members. With B2DROP, we can securely store and organize our project documents, share files with ease, and collaborate in real-time on documentation such as deliverables. The platform's emphasis on data security and privacy ensures that our files are protected and accessible only to authorized individuals, meeting the strict standards required for research data management. By leveraging the capabilities of B2DROP, we can enhance our file storage, sharing, and collaboration processes, facilitating efficient and productive collaboration within the EXTRACT project.

4.3. Standards and Guidelines

Standards and guidelines play a crucial role in ensuring consistent and standardized coding practices, particularly in a distributed software development project where teams are located in different geographical locations. By adhering to these guidelines, the EXTRACT project aims to promote cohesion, improve collaboration, and maintain code quality throughout the development process.

4.3.1. Design Patterns

Within EXTRACT, developers will use design patterns when applicable. Design patterns are time-tested solutions to recurring design problems and offer several benefits:

1) Provide solution to issues in software development using a proven solution.
2) Design patterns make communication between designers more efficient.
3) Facilitate program comprehension.

4.3.2. Code Comments

Code comments help to explain and describe the actions of a certain block of code, describing behaviours that cannot otherwise be clearly expressed in the source language and easing comprehension. EXTRACT developers will comment crucial parts in the source code to help other developers understand their code.

Despite numerous benefits of having properly commented source code, comments can be misguiding if not used properly. Thus a few points worth consideration while writing comments are:

1) Comments can get out of sync with the code if people change the code without updating the comments. Thus, comments should always change together with code.
2) Good comments are hard to write and time consuming but pay off in long term.
3) Adding comments can be counter-productive if the information provided by them is not relevant to the part of code where they are provided. Hence, inline comments should describe the next line of code.

4.3.3. Programming Style

Programming style is a set of rules or guidelines used when writing the source code. These guidelines include elements common to a large number of programming styles such as the layout of the source code, including indentation, the use of white space around operators and keywords, the capitalization of keywords and variable names, the style and spelling of user-defined identifiers, such as function, procedure and variable names; and the use and style of comments. Since the EXTRACT project will include several components that are already under development and follow their respective programming styles, developers in the frame of the EXTRACT project will follow these styles.

5. Conclusion

This deliverable is part of EXTRACT project´s WP4. Within this work package, the software layer that deals with the compute continuum is defined, designed and implemented, featuring a common abstraction to deploy and schedule data-mining workloads. The development of cybersecurity mechanisms to ensure a secure execution across this compute continuum is also a primary objective of this work package. Task 4.1 focuses on the specification of requirements related to the compute continuum, in tight collaboration with other technical work packages. In this document, the edge-to-cloud continuum architecture is described in detail, later to define the requirements that this platform should fulfil, divided into edge, HPC, cloud and security requirements. Finally, an initial version of the integration plan of the EXTRACT Platform is presented.

As documented in this deliverable, it can be stated that the objectives set for the initial milestone of task T4.1 have been accomplished successfully. Future works in the context of WP4 will include the definition of the programming and execution models' interoperability, as well as the development and implementation of all the cybersecurity functionalities that have been described in this deliverable.

6. Acronyms and Abbreviations

- AI – Artificial Intelligence
- API – Application Programming Interface
- AWS – Amazon Web Services
- CA – Consortium Agreement
- 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)
- EB – Executive Board
- EC – European Commission
- FPGA – Field-Programmable Gate Array
- GA – General Assembly / Grant Agreement
- GCP – Google Cloud Platform
- GPU – Graphics Processing Unit
- GUI – Graphic User Interface
- HPC – High Performance Computing
- IDE – Integrated Development Environment
- IoT – Internet of Things
- IPR – Intellectual Property Right
- ISO – International Organization for Standardization
- KPI – Key Performance Indicator
- K3s – Lightweight Kubernetes
- K8s - Kubernetes
- LAN - Local Area Network
- M – Month
- MS – Milestones
- NAT – Network Address Translation
- NIST – National Institute of Standards and Technology
- OCI – Oracle Cloud Infrastructure
- OS – Operation System
- OWASP – Open Web Application Security Project
- 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
- TCP – Transmission Control Protocol
- TLS – Transport Layer Security
- UDP – User Datagram Protocol
- VCS – Version Control System

- VM – Virtual Machine
- VPN – Virtual Private Network
- WP – Work Package
- WPL – Work Package Leader

7. 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.ibm.com/cloud/object-storage
[7] https://www.redhat.com/es/technologies/storage/ceph
[8] https://kubernetes.io/
[9] https://skypilot.readthedocs.io/en/latest/
[10] National Institute of Standards and Technology. (2021). *"Data Integrity: Identifying and Protecting Assets Via Enterprise Resilience Guidebook"* (NIST Special Publication 1800-25)
[11] https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp2089.pdf
[12] National Institute of Standards and Technology. (2012). *"Recommendation for Applications Using Approved Hash Algorithms"* (NIST Special Publication 800-107)
[13] https://pages.nist.gov/800-63-3/sp800-63b.html
[14] https://github.com/kubernetes-retired/kubefed
[15] https://k3s.io/
[16] https://www.redhat.com/en/technologies/cloud-computing/openshift
[17] https://www.rancher.com/
[18] https://www.kubeflow.org/
[19] https://microk8s.io/
[20] https://kubeedge.io/en/
[21] https://nuvla.io
[22] https://www.okd.io/