2

General Description

i3-MARKET leverages on blockchain technologies (e.g. Hyperledger and Ethereum) to build a trusted, interoperable, and decentralized substrate (backplane) allowing to create a federated data market where data spaces and marketplaces are able to trade data assets among each other. The i3-MARKET is mostly a set of independent subsystems with a self-contained functionality such as the identity and access management system, the semantic engine subsystem, data access subsystem, etc. Most of these subsystems have broken down their functionality into atomic and loosely coupled components exposing their functionality through a REST API, which yields a microservice-based nature to the i3-MARKET system

2.1 Deployment and Operational Concepts

Help to choose the right technologies to be used:
Choosing the right technologies for software deployments can be a complex process, but here are some general guidelines to help you make informed decisions:

2.1.1 Consider the requirements of the software

The first step in choosing the right technologies for a deployment is to consider the requirements of the software being deployed. This includes factors such as the operating system, the programming language used, the database management system, and any dependencies or third-party libraries required.

2.1.2 Evaluate the deployment environment

The deployment environment will also play a key role in determining the appropriate technologies to be used. Consider factors such as the hardware and software infrastructure, the network configuration, and the security requirements.

2.1.3 Consider automation and orchestration

Automation and orchestration tools can help to streamline the deployment process and minimize the risk of errors or inconsistencies. Consider using tools such as Ansible, Chef, or Puppet to automate the deployment process.

2.1.4 Evaluate containerization options

Containerization technologies such as Docker and Kubernetes can help to simplify the deployment process and make it more portable across different environments. Consider using containerization technologies to deploy software in a consistent and repeatable way.

2.1.5 Consider monitoring and reporting tools

Monitoring and reporting tools can help to ensure that the software is performing as expected and can alert teams to potential issues before they become critical. Consider using tools such as Nagios, Prometheus, or Grafana to monitor and report on key metrics.

2.2 Deployment Specification

The i3-MARKET architecture specification is based on the 4 + 1 architectural view model approach. One of these views, physical view, is the scope of this document. Physical view depicts the system from a system engineer's point of view. It concerns the topology of software components on the physical layer as well as the physical connections between these components. This view is also known as the deployment view. UML diagrams used to represent the physical view must include the deployment diagram.

Considering this in the i3-MARKET context, the deployment specification should define execution architecture of systems that represent the assignment (deployment) of software artifacts (i3-MARKET building blocks) to deployment targets (usually nodes).

Nodes represent either hardware devices or software execution environments. They could be connected through communication paths to create network systems of arbitrary complexity. Artifacts represent concrete elements in the physical architecture.

Once the deployment has been provided, a complementary specification would be necessary to define how to deploy software within the i3-MARKET ecosystem. In the context of i3-MARKET, we will be referring to this specification as management operative specification.

Finally, an end-user operative specification is provided, defining the interaction with i3-MARKET from a stakeholder point of view.

2.3 Terminology

The key terms behind i3-MARKET deployment terminology are the following:

Artifact:
As it is described in [1], an artifact is a classifier that represents some physical entity, a piece of information that is used or is produced by a software development process, or by deployment and operation of a system. Artifact is a source of a deployment to a node. A particular instance (or "copy") of an artifact is deployed to a node instance. The most common artifacts are as follows:

- Source files
- Executable files
- Database tables
- Scripts
- DLL files
- User manuals or documentation
- Output files

Artifacts are deployed on the nodes. They can provide physical manifestation for any UML element. Generally, they manifest components. Artifacts are labelled with the stereotype <<artifact>>, and it may have an artifact icon on the top right corner.

Each artifact has a filename in its specification that indicates the physical location of the artifact. An artifact can contain another artifact. It may be dependent on one another.

Artifacts have properties and behaviour that manipulate them.

Node:
As it is introduced in [2], a node is a computational resource upon which artifacts are deployed for execution. A node is a physical thing that can execute one or more artifacts. A node may vary in its size depending on the size of the project.

Node is an essential UML element that describes the execution of code and the communication between various entities of a system. It is denoted by a 3D box with the node name written inside of it. Nodes help to convey the hardware that is used to deploy the software.

An association between nodes represents a communication path from which information is exchanged in any direction.

Generally, a node has two stereotypes as follows:

- **<< device >>:** It is a node that represents a physical machine capable of performing computations. A device can be a router or a server PC. It is represented using a node with stereotype <<device>>. In the UML model, you can also nest one or more devices within each other.
- **<< execution environment >>:** It is a node that represents an environment in which software is going to execute. For example, Java applications are executed in Java virtual machine (JVM). JVM is considered as an execution environment for Java applications. We can nest an execution environment into a device node. You can nest more than one execution environments in a single device node.

The following sections report on the deployment strategy and the status reached at the closure of the final release.

2.4 i3-MARKET Artifacts Overview

In the context of i3-MARKET, several artifacts have been developed, integrated, and deployed. These artifacts have been built on top of a set of third-party and open-source frameworks, which have been analysed and deployed as tech-bed for the construction of the i3-MARKET backplane. For the final release, the third-party artifacts included on i3-MARKET are:

- Hyperledger Besu: The blockchain framework.
- CockroachDB: Distributed database deployed on each node. Admin Interface only accessible through node 1.
- RocksDB: Decentralized storage included with the blockchain network (ledger).
- Loopback4: Framework supporting i3-MARKET backplane API.

Regarding the project-internal conceptual artifacts, i3-MARKET has developed an extensive artifacts portfolio, mainly provided in WP3 and WP4, for supporting the integration, registration, discovery, and transfer of reliable trade of data. A detailed list of these artifacts (including artifact ID, artifact name, artifact dependencies, and their status for the final release) can be seen in Table 2.1.

Table 2.1 i3M proprietary conceptual artifacts.

Artifact ID	Artifact	Dependencies	Final release use	Notes
A1	Blockchain framework	Decentralized storage	Deployed and used	Blockchain framework. Deployed on each node.
A2	CockroachDB (distributed storage)		Deployed and used	Distributed database deployed on each node.
A3	Decentralized storage	Blockchain framework	Deployed and used	Included with the blockchain framework.
A4	User-centric authentication		Deployed and used	Each instance/pilot has its own OIDC and VC service.
A5	Service-centric authentication		Deployed and used	Each instance/pilot has its own Keycloak service.
A6	HW Wallet		In progress	
A7	Software Wallet	Cloud Wallet Client, Backplane API (Cloud Wallet server and user-centric authentication), data access SDK, and i3-MARKET SDK	Deployed and used	
A8	Smart contract manager	SLA/SLE smart contract	Deployed and used	
A9	SLA/SLE smart contract		Deployed and used	
A10	Conflict resolution	SCM and DS	Deployed and used	Integrated with Besu, smart contract manager and decentralized storage. Each instance/pilot has its own service.
A11	Explicit user consent	Backplane API (smart contract manager, distributed ledger, and distributed storage)	Deployed and used	Integrated with the smart contract manager.
A12	Auditable accounting		Deployed and used	
A13	Standard payment	Backplane API (auditable accounting, conflict resolution, smart contract, and SLA/SLE smart contract)	Deployed and used	Library to be integrated and deployed in data access SDK and data access API. Library for the i3-MARKET non-repudiation protocol that helps generate/verifying the necessary proofs and the received block of data.
A14	Tokenization	Backplane API (user-centric authentication, smart contract, and SLA/SLE smart contract)	Deployed and used	

Table 2.1 *Continued.*

Artifact ID	Artifact	Dependencies	Final release use	Notes
A15	Micro payment		Deployed	Integrated into the Tokenizer. Low chance to be used by i3-MARKET because for data payments is used fiat money and the Tokenizer and the token are just for the fees.
A17	Data access API		Deployed and used	Each node has its own endpoint.
A18	Semantic data manager (triple store)		Deployed and used	
A19	Semantic models		Deployed and used	It is not software component.
A20	Semantic engine	Backplane API (user IDs) and decentralized storage	Deployed and used	This component includes - Semantic model management - Offering and discovery Each instance/pilot has their own engine
A21	Backplane API	All	Deployed and used	Each node has its own endpoint
A22	i3-MARKET SDK-generator		Deployed and used	Endpoint at node 1 Deployed as Docker container through Ansible
A26	SDK-RI (reference implementation)	All	Deployed and used	Each marketplace has its own SDK-RI
A27	SDK-core	SDK-generatore All	Deployed and used	Available at Nexus
A29	Secure server (Keycloak)		Deployed	Available at Nexus Integration with user-centric authentication component in progress
A30	Notification manager	SDK-RI and SDK-core	Deployed and used	
A31	Rating		Deployed and used	

Finally, in the context of CI/CD, a set of tools has been used for the automation and monitoring of the artifacts deployed on i3-MARKET. These tools are listed in the deliverable D4.7 and in the sections below.

2.5 Deployment Architecture View

The i3-MARKET deployment view is depicted in the picture below. Four nodes constituted the i3-MARKET R1 cluster. On each node, it will be deployed a Backplane gateway system and an instance of all the rest i3-MARKET main building blocks (trust, security, and privacy system, storage system, and data access system) giving backend support to the Backplane gateway system. In addition to that, node 4 will host all the components related with the semantic engine building block.

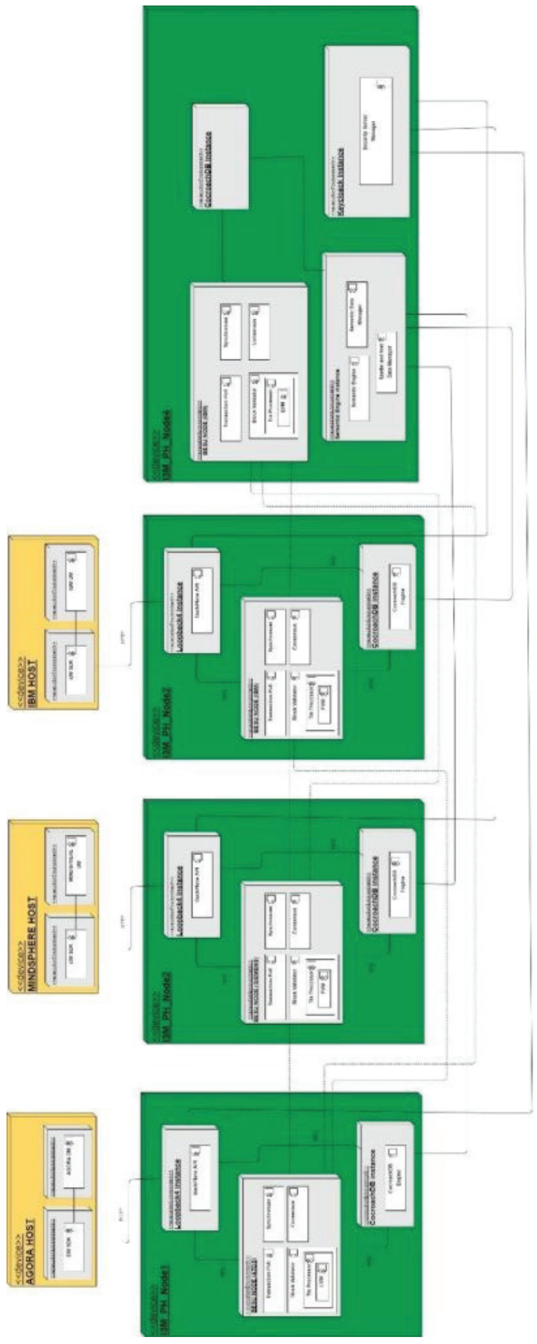

Figure 2.1 i3M ecosystem deployment diagram.

2.6 i3-MARKET Network Infrastructure

Figure 2.1 shows the deployment diagram associated with the i3-MARKET network for the last release. It can be appreciated that the deployment strategy has evolved from the M18 centralized infrastructure (where a single and centralized i3-MARKET instance gave support to all demonstrators) to a "hybrid" decentralized infrastructure (where each of the pilot's demonstrator that joined the i3-MARKET ecosystem has its own i3-MARKET instance). It is important to highlight the "hybrid" nature of the network because a master instance is maintaining, among other reasons, some centralized services such as the central Besu node, the notification manager, etc., and CI/CD tools needed for the setup of the network.

Therefore, in this landscape, it can be appreciated the existence of marketplaces, which are simple instances (yellow boxes) and the central/master instances (green boxes). The most significant relationship among the instances is the connection between each of the Besu nodes themselves and their connection with the Besu central node.

It is important to mention that the number of nodes used for each of the i3-MARKET pilot instances and the maintenance of these nodes is up to the pilots' criteria and responsibility. Thus, the node's layout that appears on each of the instances, depicted for hosting the i3-MARKET artifacts, Figures 2.1 and 2.2, is just an example and does not have to be the real picture of the instances deployment.

2.7 Software Stack

For the final release, two types of software environments (understood as a set of artifacts) can be found in i3-MARKET, which are aligned with the infrastructures presented in the previous section. On one hand, the marketplace-side software stack (i3-MARKET pilot environment) and, on the other hand, the stack landscape deployed in the centralized cluster (i3-MARKET master environment), which acts as a master for the rest of marketplaces, adhere to the i3-MARKET network.

A four-layer stack has been defined for i3-MARKET (Figure 2.3): at the lowest layer, there is the Cloud provisioning and management layer. On top of that, a DevOps software layer is placed for assembling all the software used for the CI/CD process. Then, a third-party software layer is in charge of giving support to the i3M Core Artifacts, which can be found at the top level of the stack.

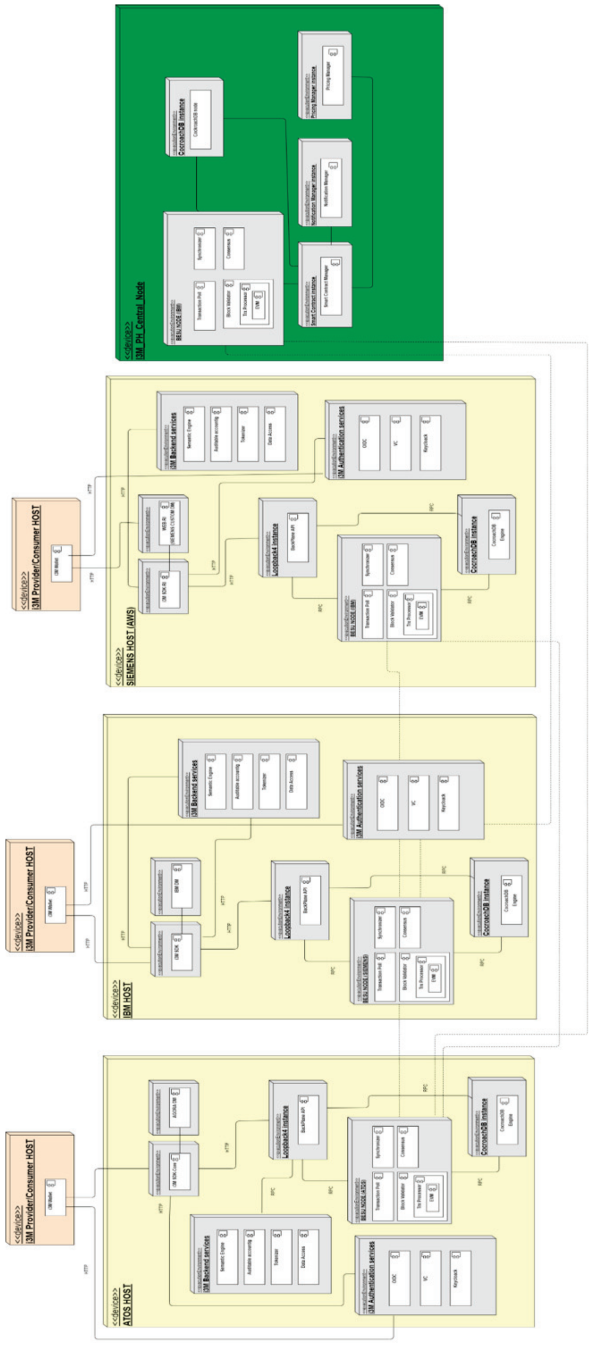

Figure 2.2 i3M ecosystem deployment diagram.

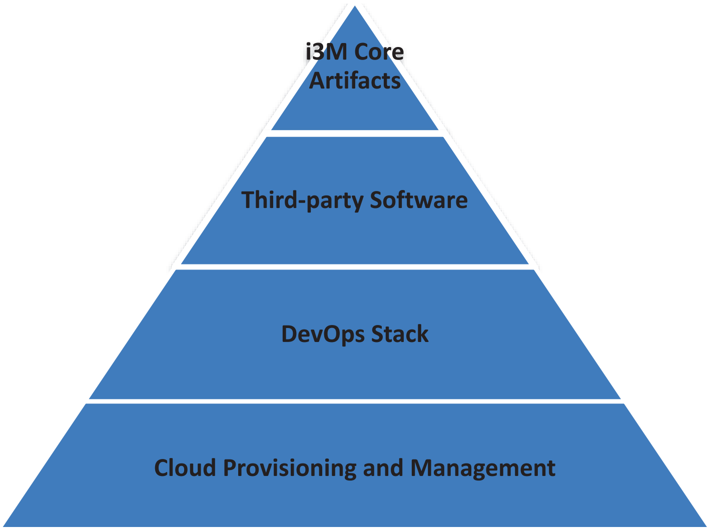

Figure 2.3 i3M SW stack four layers.

Depending on the environment to be deployed, it might deploy one layer or another. More details on the specific software deployed on each environment are given in the following sub-sections.

2.8 i3-MARKET Master Environment

The i3-MARKET centralized software stack, represented in Figure 2.4, is focused on providing the minimum and centralized services for erecting an i3-MARKET network; these are the "Cloud provisioning and management" layer, the "DevOps software" layer, master nodes of the "Third-party software" layer, and the centralized i3-MARKET artifacts provided in the "i3M centralized services" layer.

Cloud provisioning and management layer oversees providing and managing all physical nodes that the i3-MARKET common infrastructure is composed of. For the management of physical resources in a homogeneous way, an Ansible Tower[1] instance is deployed for the administration of said physical resources, thus having their management centralized from Ansible. On the other hand, for the monitoring and registering of the status of the i3-MARKET central services, Zabbix is deployed as part of the central

[1] Ansible Tower: https://www.ansible.com/products/tower

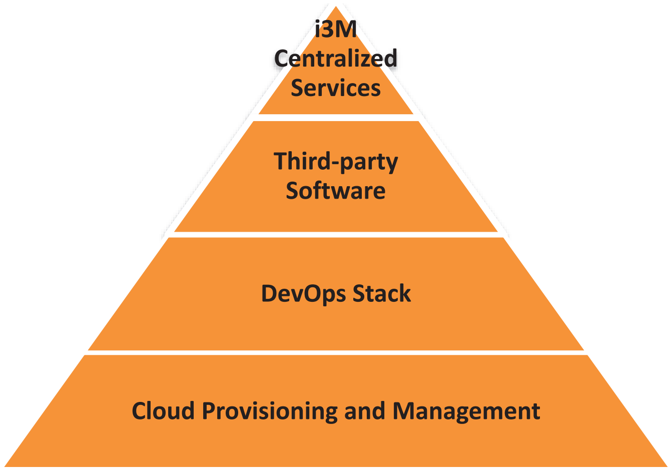

Figure 2.4 i3M centralized software stack layers.

environment. Table 2.2 shows some deployment aspects of the previously commented tools:

Table 2.2 i3M centralized cloud management and monitoring software.

SW Component	Building block	Assigned VM/PR	Type	Technology
Ansible AWX	Deployment	I3M-PH-Node2	Third-party SW	Ansible AWX
Zabbix	Monitoring	I3M-PH-Node4	Third-party SW	Zabbix

i3-MARKET DevOps will be a set of practices that will combine software development and IT operations, and it will aim to shorten the i3-MARKET system development life cycle and provide continuous delivery with high software quality. Thus, the DevOps layer combines software development and IT operations by means of the artifacts listed in Table 2.3.

Besides that, a set of artifacts from the i3-MARKET third-party software is needed in the centralized environment to master some services:

- Master Besu node, which gives authorization to new member to the blockchain network.
- Cockroach data base, which hosts the “Seed Index” for federating queries.
- RocksDB, which is the central instance of the blockchain.
- Security services for allowing authentication and authorization capabilities to the central node.

Table 2.3 i3M centralized DevOps software.

SW Component	Building block	Assigned VM/PR	Type	Technology
Ansible AWX	Deployment	I3M-PH-Node2	Third-party SW	Ansible AWX
Docker Swarm	Deployment	I3M-PH-Node1, I3M-PH-Node2, I3M-PH-Node3, and I3M-PH-Node4	Third-party SW	Docker Swarm
GitLab CI/CD (Runners)	CI/CD	GitLab (out of i3M cluster)	Third-party SW	GitLab
Nexus	CI/CD	I3M-PH-Node4	Third-party SW	Nexus
NGINX	Management/security	I3M-PH-Node1, I3M-PH-Node2, I3M-PH-Node3, and I3M-PH-Node4	Third-party SW	NGinx
MkDocs	Documentation	I3M-PH-Node4	Third-party SW	MkDocs

Table 2.4 shows some deployment details regarding the before commented artifacts.

Table 2.4 i3M centralized third-party software.

SW Component	Building block	Assigned VM/PR	Type	Technology
Blockchain framework (central node)	Blockchain network	I3M-PH-Node4	Third-party SW	Hyperledger Besu
Distributed storage	Data storage	I3M-PH-Node4	Third-party SW	CockroachDB
Decentralized storage	Data storage	I3M-PH-Node4	Third-party SW	RocksDB
Security server	Trust, security, and privacy	I3M-PH-Node4	Third-party SW	OIDC, VC, and Keycloak

Finally, regarding the "i3-MARKET centralized services", the notification manager and the SDK-generator (which support the SDK-core generator) have been designed to be centralized. Table 2.5 shows some deployment details of them.

Table 2.5 I3M centralized proprietary software.

SW Component	Building block	Assigned VM/PR	Type	Technology
Notification manager	Data storage	I3M-PH-Node4	i3-MARKET SW	RabittMQ
SDK-generator	Reference implementation	I3M-PH-Node4	Hybrid artifact	OpenAPI Generator[2]

2.9 i3-MARKET Pilot Environment

The i3-MARKET pilots' stack is represented in Figure 2.5 and it is composed mainly of two layers: "Third-party software" layer and "i3M core services" layer.

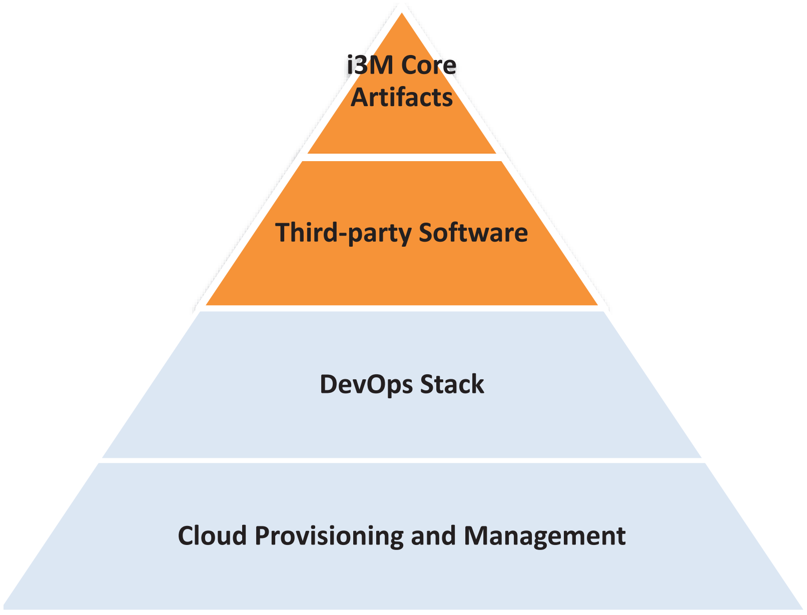

Figure 2.5 i3M pilots' software stack layers.

The top layer is composed of all i3-MARKET core artifacts supplied by the project, which might be deployed in a decentralized way. In other words, each marketplace willing to be part of the i3-MARKET ecosystem might have one instance of these artifacts running on its own i3-MARKET infrastructure. Table 2.6 shows more information about these artifacts/components as well as the set of services provided by each of them (linked with the Microservices View in D2.4). Other details that can be found in the table are:

- SW artifact/component name
- Associated building block (see internal deliverable I2.41 [3])
- Artifact type
- Technology supporting artifact

Table 2.6 i3M pilots' core artifacts.

SW Component	Building block	Services	Type	Technology
User-centric authentication	Trust, security, and privacy	Verifiable Credential API	i3-MARKET SW	Keycloak
Service-centric Authentication	Trust, security, and privacy	OIDC provider API	i3-MARKET SW	
Cloud Wallet	Trust, security, and privacy	Wallet Cloud Server and Wallet APP	i3-MARKET SW	
HW Wallet	Trust, security, and privacy		i3-MARKET SW	
Smart contract manager	Trust, security, and privacy	Smart contract manager API + explicit user consent	i3-MARKET SW	Hyperledger Besu, Solidity
Conflict resolution	Trust, security, and privacy	Conflict resolution API	i3-MARKET SW	
Auditable accounting	Trust, security, and privacy	Auditable accounting API	i3-MARKET SW	
Monetization	Trust, security, and privacy	Pricing manager API, Tokenizer API, and non-repudiation protocol library	i3-MARKET SW	
Data access	Data access	Data access API, standard payments system, and data transfer	i3-MARKET SW	

Table 2.6 *Continued.*

SW Component	Building block	Services	Type	Technology
Semantic	Semantics	Semantic engine API (metadata registry management, data offerings, and federated query discovery)	i3-MARKET SW	MongoDB
Backplane API	Backplane		i3-MARKET SW	LoopBack4
SDK-RI	Reference implementation		i3-MARKET SW	Java
Web-RI	Reference implementation		i3-MARKET SW	

Finally, the "Third-party SW" layer will be mainly in charge of providing the software stack identified as software requirements by the i3-MARKET system. These software requirements are: Hyperledger Besu, CockroachDB, Loopback4, and Keycloak. The Table 2.7 summarise the i3M pilot third party artifacts used.

Table 2.7 i3M pilots' third-party artifacts.

SW Component	Building block	Type	Technology
Blockchain framework	Blockchain network	Third-party SW	Hyperledger Besu
Distributed storage	Data storage	Third-party SW	CockroachDB (deployed standalone)
Decentralized storage	Data storage	Third-party SW	RocksDB
Security server	Trust, security, and privacy	Third-party SW	Keycloak

Regarding "DevOps Stack" and "Cloud provisioning and management", these two layers are out of scope of the stack provided by i3-MARKET on each external instance. This is mainly because of two reasons:

- Each pilot is responsible for deciding, deploying, and using the nodes management and service monitoring tools most suitable to its needs and restrictions. Thus, for example, IBM pilot has decided to use Trivy[3] for scanning vulnerabilities in the deployment of its i3-MARKET instance.

- As it was commented in the infrastructure sections, self-management by the pilot is assumed where to deploy each artifact. Therefore the "Cloud provisioning and management" layer is now under the scope of the pilot administrators.

[3] https://www.aquasec.com/products/trivy/