REPLY

# BLEDGER – DIGITAL PLATFORM FOR COLLABORATIVE DATA-COLLECTION IN BUSINESS NETWORKS

## TECHNICAL ANALYSIS

OCTOBER, 2025

Bocconi

# DOCUMENT GOAL

Goal

This document provides an overview of the blockchain-based application BLedger, outlining its process flow and high-level architecture. It explains the purpose of each component – differentiating between private data and digital assets – and describes the infrastructure for wallet management, including the use of multi-signature wallet technology. It also details how tokens are used within business networks, highlights the flexibility of editable metadata, and defines the standard data elements shared across all participants.

# AGENDA

1. **Process overview**
2. High-level architecture
3. Bledger Network Token
4. Blockchain benefits and transparency
5. Verify data on-chain
6. Bledger screenshots

# THE ACTORS

The platform designs a value chain operated by the following actors:

**Actors**


Business Unit (organization)


Business Network Administrator


Third-party auditors

- **Business Units (BUs):** Corporations, SMEs and startups using the platform to enhance data sharing with business network partners
- **Business Network Administrators (BNAs):** The entity that is in charge for relevant administrative tasks for the network.The business unit that proposes the creation of a business network automatically becomes the business network administrator of that network
- **Third-party auditors:** External entities that provide independent validation of the data shared on the platform.

# THE PROCESS

The platform is designed to handle a structured workflow that ensures seamless interaction between different entities while maintaining security, transparency, and efficiency.

**Process**


- Pre-registered company
- Wallet assignment

- Business Network creation
- Business network management

- Data upload and contextual format verification (csv)
- Key data extraction

- Data is extracted to calculate the scope 3

- Third-party auditors access the platform and verify data

- Once the information has been extracted, the business units can download the data to calculate their scope 3 carbon emissions

# THE PROCESS - FOCUS ON ONBOARDING

Onboarding

This streamlined solution represents the approach implemented for the MVP, with the aim of enabling the simulation of the various roles within the platform

Pre- registered Company

The different roles will be pre-registered, and each role will be assigned functionalities to operate on the platform in accordance with its specific role

| Actor | Description | Functionalities |
|---|---|---|
| **Business Unit (BUs)** | Corporations, SMEs and startups using the platform to enhance data sharing with business network partners | • Participate to business network<br>• Upload data for Scope 1 e Scope 2<br>• Retrieve data for Scope 3 calculation |
| **Business Network administrators (BNAs)** | The entity that is in charge to perform relevant administrative tasks for the network | • Same functionalities of the BUs, plus<br>• Manage business networks and hierarchy |
| **Third-party auditors (TPAs)** | External entities that provide independent validation of the data shared on the platform. | • Certify the data uploaded by the different companies within the network |

# THE PROCESS - FOCUS ON BUSINESS NETWORK


A Business Unit requests the creation of a business network, selecting the companies to include in the network and the third-party auditors

The business unit that proposes the creation of a business network automatically becomes the business network administrator of that network

At any time, the business network admin can update the business network (es. Add/ remove participant, add third-party auditors)

The companies who receive the invitation must accept or refuse to join the network

The Business network administrator, builds the hierarchy within the network, ensuring that information within the network can be viewed by those who have the right to do so. This hierarchy is then reflected in a smart contract for the management of the data flow

# THE PROCESS - FOCUS ON DATA MANAGEMENT

Upload and data management | Extraction

Document upload | Key data Extraction | Data hash storage on blockchain | Data visualization | Scope 3 Calculation

A business unit uploads 3 CSV files. For the Scope 1 & Scope 2 documents the platform verifies the correctness of the format before completing the upload.

The documents are then matched with the metadata file.

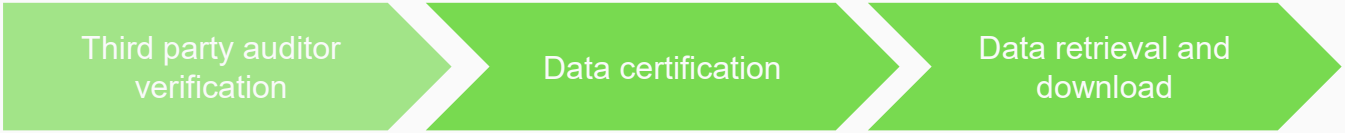
From the Scope 1 & Scope 2 CSV file document, the most significant data are extracted in order to retrieve the main component for Scope 1 and Scope 2 emission

Once the data are extracted, the hash is calculated and recorded on the blockchain. The hash is clickable and allows direct access to the transaction on the blockchain (Polygon)

Once the data have been extracted and the hash stored on blockchain, they are visualized and made available according to the hierarchy defined by the BNA, so that each actor can view the data pertinent to their responsibilities

Once all the information has been retrieved and the additional required information has been entered, the company can proceed to calculate its Scope 3 on the platform

# THE PROCESS – DATA CERTIFICATION AND RETRIEVAL

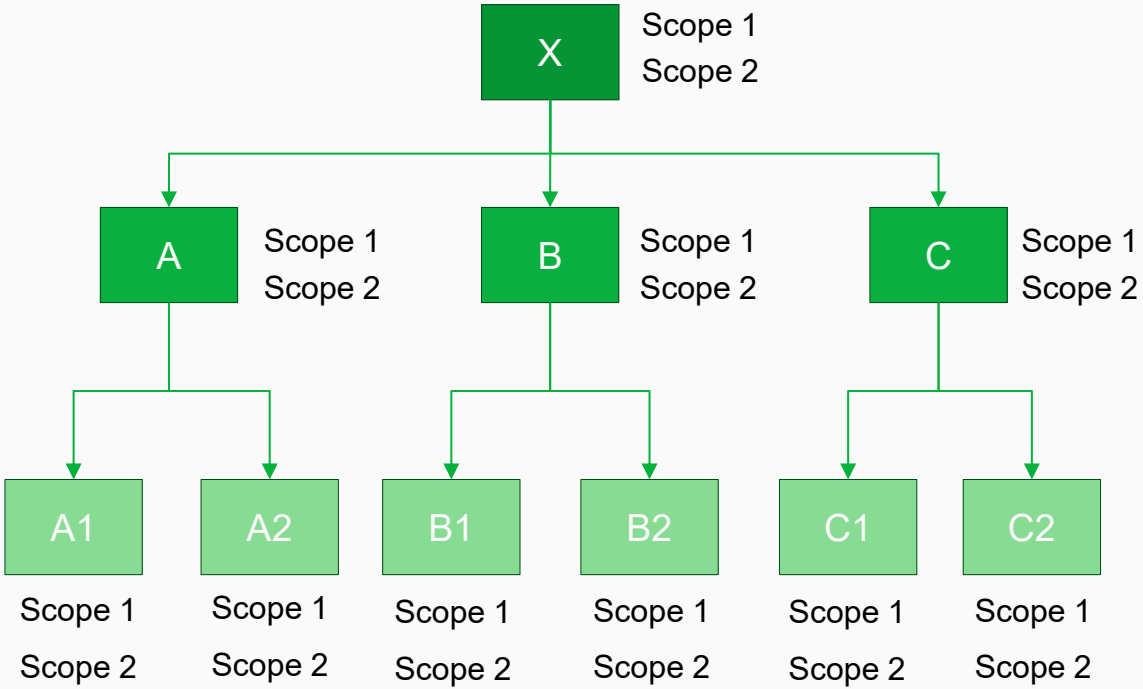

Once the data have uploaded, the business unit can request the verification from a third-party auditors in the business network. TPAs certify their correctness by verifying that the Scope 1 and Scope 2 calculations have been performed correctly

Once the data have been certified, a checkmark will appear next to the data to show it is certified. The verification is tracked on the blockchain

Once all the relevant business units uploaded the data, the companies can download the information from their supplier and customer, in order to gather the information to calculate autonomously their own Scope 3.

# FOCUS ON BUSINESS NETWORK “HIERARCHY” PRODUCT-BASED

Data retrieval




Each actor uploads information regarding their **Scope 1 and Scope 2**, along with relevant details, into the platform to enable other actors to calculate their Scope 3

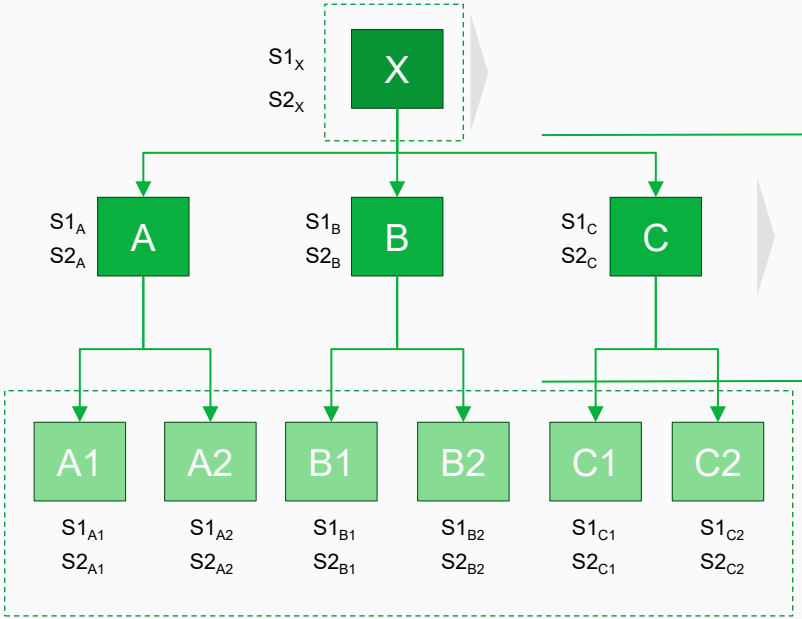
# VARIABLES TO BE INSERTED / CALCULATED FOR SCOPE 3

| Variables | Description | Inserted by | Calculated as |
|---|---|---|---|
| Ei | Per−node cradle−to−gate emission intensity, E represents the emission intensity, per unit of volume produced by the node i, useful for the Scope 3 calculation | n.a. | Calculated by node i |
| $Q_{ij}$* | It represents the quantities produced by node *j*, which are used for the product made by node *i* | Inserted by node i | n.a. |
| $T_{i \rightarrow j}$ | It represents the emissions from transporting products from supplier *j* to customer *i* | Inserted by node i | n.a. |
| Production $volume_i$ | It represents the volume produced by node *i* | Inserted by node i | n.a. |

* In this formula i is the client and j is the supplier and $Scope3_i = ProductionVolume_i \times E_i$

# FOCUS ON BUSINESS NETWORK “HIERARCHY” PRODUCT-BASED

Data retrieval




$$E_X = (S1_X + S2_X) + q_{XA}\,(\boldsymbol{E_A} + t_{A\to X}) + q_{XB}\,(\boldsymbol{E_B} + t_{B\to X}) + q_{XC}\,(\boldsymbol{E_C} + t_{C\to X})$$

$$E_A = (S1_A + S2_A) + q_{AA_1}\,(\boldsymbol{E_{A_1}} + t_{A_1\to A}) + q_{AA_2}\,(\boldsymbol{E_{A_2}} + t_{A_2\to A})$$

$$E_B = (S1_B + S2_B) + q_{BB_1}\,(\boldsymbol{E_{B_1}} + t_{B_1\to B}) + q_{BB_2}\,(\boldsymbol{E_{B_2}} + t_{B_2\to B})$$

$$E_C = (S1_C + S2_C) + q_{CC_1}\,(\boldsymbol{E_{C_1}} + t_{C_1\to C}) + q_{CC_2}\,(\boldsymbol{E_{C_2}} + t_{C_2\to C})$$

$$E_{A_1} = (S1_{A_1} + S2_{A_1}) \qquad E_{A_2} = (S1_{A_2} + S2_{A_2})$$

$$E_{B_1} = (S1_{B_1} + S2_{B_1}) \qquad E_{B_2} = (S1_{B_2} + S2_{B_2})$$

$$E_{C_1} = (S1_{C_1} + S2_{C_1}) \qquad E_{C_2} = (S1_{C_2} + S2_{C_2})$$

The following **formula** is then used for **Scope3 calculation:** $Scope3_i = ProductionVolume_i \times E_i$

# AGENDA

1. Process overview
2. **High-level architecture**
3. Bledger Network Token
4. Blockchain benefits and transparency
5. Verify data on-chain
6. Bledger screenshots

# HIGH LEVEL ARCHITECTURE

Bledger architecture pillars

The platform is built to be used in enterprise contexts. The implementation is done using only open software stacks and framework and open standards to be able to swap components (middleware, EVM blockchain etc.)

## Web user interface

The **Web UI** approach allows broader access without technical requirements other than a device with a **modern browser**.

The web app is not requiring bearer wallet provider injection to comply with broader **enterprise grade web application standards**

## Backend middleware

The platform's hearth is based on a backend that implements **core logics**, wallet **integration** and management, smart contract **interaction** and private data and **workflows** management.

It is the core service that connects web ui to a private database to the blockchain in a scalable and secure solution.

## Private storage

The solution implements a private and flexible storage layer critical for privacy and security constraints for enterprise grade and regulatory compliance application implementation.

Thanks to this solution **blockchain data** can be integrated with **private data** allowing **permissioned** access to original raw data.

## Blockchain

The blockchain ledger and smart contracts are the foundation for **immutable** and **verifiable** data certification, **interoperability** and **auditability**.

The adoption of a **public blockchain** enables native interoperability of the platform with external solutions.

# HIGH LEVEL ARCHITECTURE

Bledger architecture – Components

## Database

The application uses **PostgreSQL** as its database, a powerful open-source relational database system known for its **reliability**, **scalability**, and advanced support for complex queries and **data integrity**.

## Backend

The backend of the application is built with **Spring Boot**, leveraging **Spring WebFlux** to handle asynchronous and non-blocking operations.
This architecture ensures high **performance** and **scalability**, allowing the system to efficiently manage multiple requests and real-time **data processing** across the platform.

## Frontend

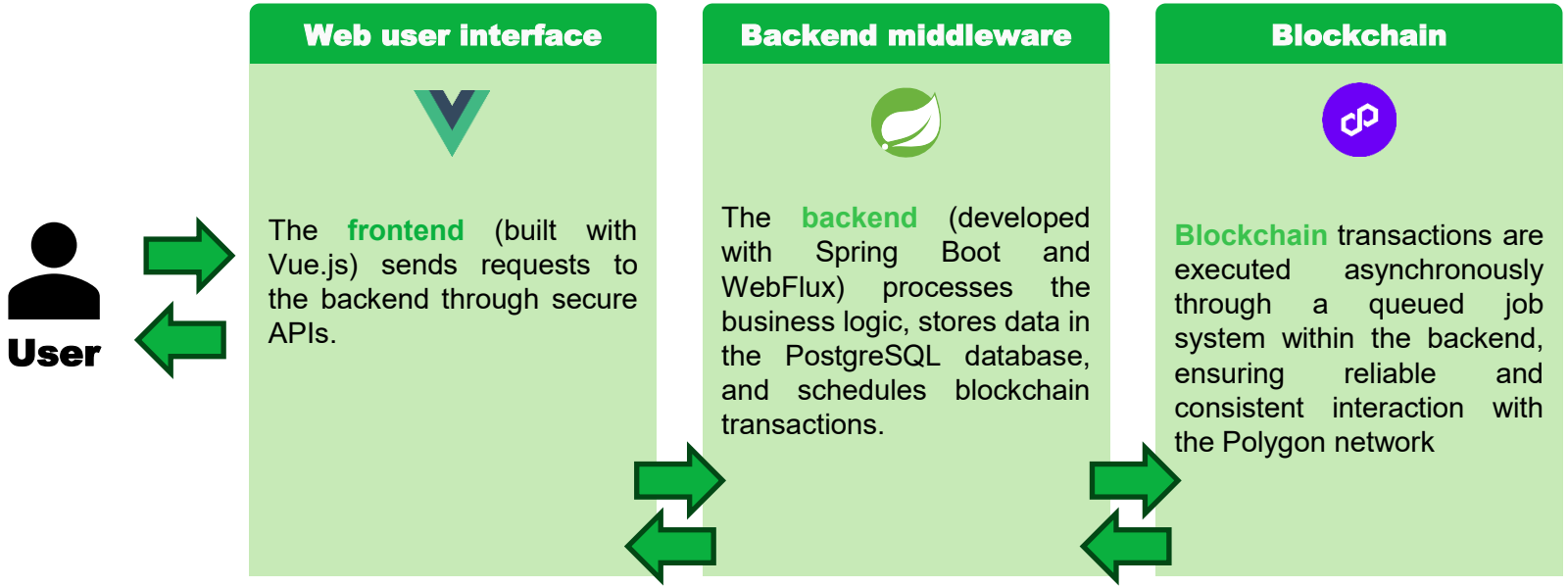
The frontend of the application is developed using **Vue.js**, a modern JavaScript framework for building user interfaces.
It provides a responsive and dynamic experience, allowing users to interact seamlessly with the platform through an intuitive and efficient web interface.

## Blockchain

For the blockchain layer, the application leverages the **Polygon** network, ensuring fast and low-cost transactions.
The smart contracts are developed in **Solidity**, enabling secure and transparent management of data and interactions on the blockchain.

# HIGH LEVEL ARCHITECTURE: DATA FLOW

Bledger architecture – System components interactions



# ARCHITECTURE: DEPLOYMENT MODELS

Bledger is designed for scalability and security within

The implementation adopts the **container-based** approach designing each component as a single container with its own **DevOps** pipeline for building, testing. This implementation is wanted also to **reduce any specific cloud provider lock-in** unbinding the software platform from specific SaaS or PaaS models.

- ❑ **Portability** - Applications run consistently across any environment
- ❑ **Isolation** - Each container operates independently, improving security and preventing conflicts
- ❑ **Efficiency** - Lightweight and fast, using fewer resources than virtual machines
- ❑ **Scalability** - Easy to replicate and scale applications up or down quickly
- ❑ **Microservices-friendly** - Perfect for breaking applications into smaller, independent services
- ❑ **DevOps integration** - Streamlines development, testing, and deployment pipelines
- ❑ **Cost-effective** - Better resource utilization means lower infrastructure costs

In essence, containers make software more reliable, flexible, and easier to manage throughout its entire lifecycle.

**DEV**

Simple development environment setup. Easy to integrate in different development scenarios and pipelines

**OPS**

Standard and prototyped deployment scenarios for quick deployment any organization can implement

**SCALE**

Capability to quickly scale horizontally and vertically according to specific business network requirements

# ARCHITECTURE: DEPLOYMENT EXAMPLES

Showcase of some infrastructure implementations currently tested

During the development phase infrastructure deployment models have been created and tested as well. In this document two optional deployment scenarios will be shown with their pros and cons.

**Quick start / demo**

This is the most efficient method to commence implementing the solution by utilizing a single Linux virtual machine (VM) with Docker installed, along with a Docker Compose file that enables the deployment of the entire solution on a single server.
This solution can be effortlessly installed on on-premises Linux machines and virtual machines provided by any cloud service provider available in the market. The reference specifically adopts the AWS cloud provider, but this is not a mandatory requirement. The same architecture is also available on other cloud platforms, such as Azure, Google Cloud Platform (GCP), and so on.

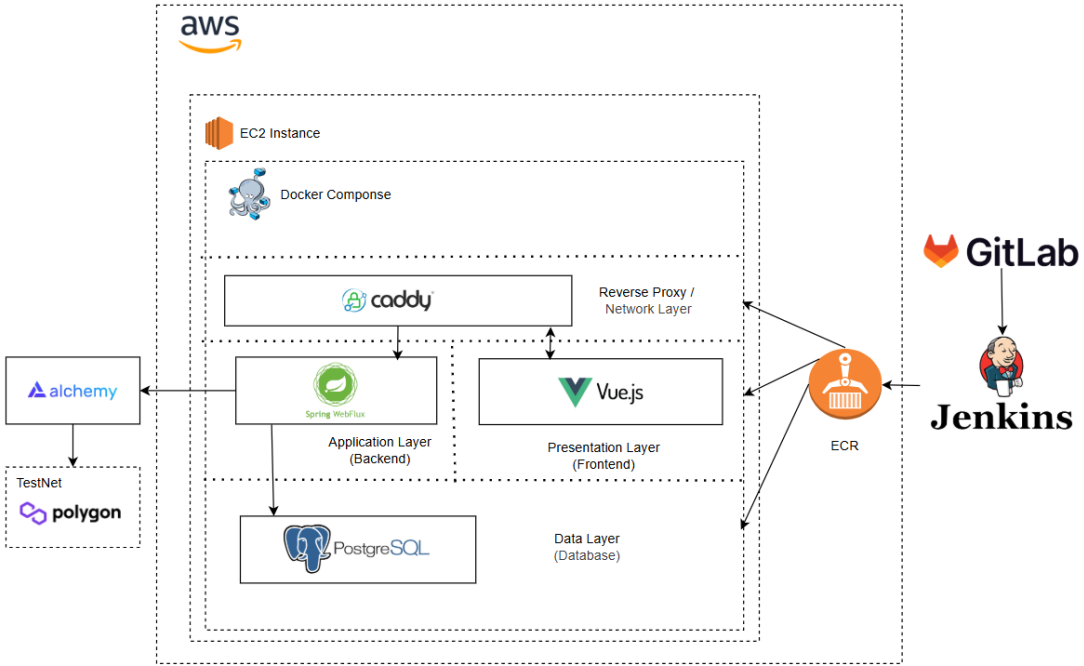
**Production**

This solution leverages the pillars of the platform on top of a cloud-based PaaS environment, utilizing production-grade infrastructure to enhance deployment resilience. It incorporates enterprise-grade cybersecurity measures for key management, scalability, multi-availability zone configurations, automated backup and monitoring tools, and more.
This solution leverages the platform's core principles while operating within a cloud-based
As per the quick start solutions, this implementation is built on top of AWS services. However, the AWS services are not mandatory: similar container-based, scalable implementations can also be achieved using any other major cloud provider.

# DEPLOYMENT MODELS | QUICK-START/DEMO

Bledger architecture – Deployment developed for demo purposes

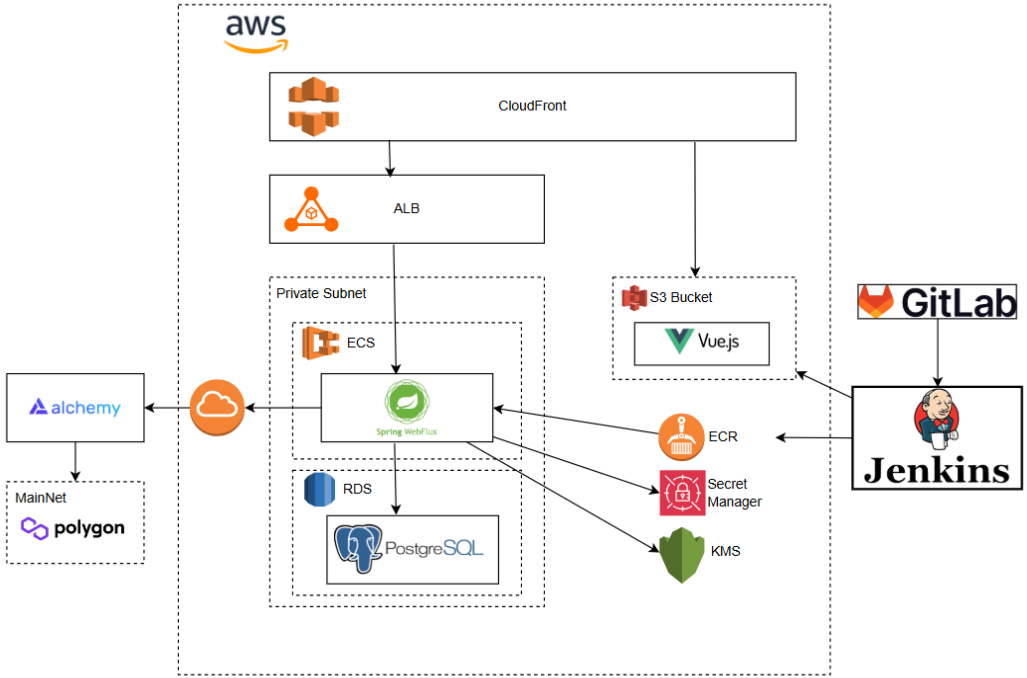
This architecture deploys a three-layer web application on an AWS EC2 instance orchestrated with **Docker Compose**: a presentation layer built with **Vue.js**, a backend application layer using **Spring Boot**, and a data layer on **PostgreSQL**. Incoming traffic is terminated and routed by a Caddy **reverse proxy**/network layer to the frontend and backend containers. The backend integrates with blockchain services via **Alchemy** and **Polygon** (TestNet) for on-chain interactions. CI/CD is handled through a GitLab-to-Jenkins pipeline that builds images and pushes them to a container registry (ECR), which are then pulled and run on the EC2 host. Overall, it provides a containerized, proxy-fronted, **CI/CD-driven** stack with external Web3 integrations.



# DEPLOYMENT MODELS | PRODUCTION

Bledger architecture – Production example on AWS Cloud

The architecture leverages CloudFront as the **content delivery network** at the **edge**, with Application **Load Balancer** (ALB) distributing traffic to backend services. The core application infrastructure utilizes Amazon ECS for **container orchestration** with underlying compute resources, alongside RDS for **relational database management** and PostgreSQL for additional data persistence, all contained within an Elastic Subnet. Infrastructure provisioning and management is handled through **infrastructure-as-code** tools including Alchemy and Polygon for Mainnet deployment, while **Terraform** orchestrates the overall infrastructure. The system integrates multiple AWS services for different concerns: **S3 buckets for object storage** (for token private metadata storage), ECR (Elastic Container Registry) for Docker image storage, Secrets Manager for secure credential management, KMS for **encryption key management**, and Vue.js (Vuejs) for the frontend application framework, creating a modern, scalable cloud-native application platform.



# DEPLOYMENT MODELS COMPARISON

Bledger architecture

This slide compares two deployment models for a Bledger architecture system: Quick-start/Demo and Production environments.

| | Pro | Cons |
|---|---|---|
| **Quick-start/Demo** | • *High flexibility*<br>• *Deploy anywhere*<br>• *Simple to set up and maintain*<br>• *Low operational costs* | • *Not scalable*<br>• *No high availability (HA)*<br>• *Non-prod grade security* |
| **Production** | • *High availability (HA)*<br>• *Enhanced security and network isolation*<br>• *Scalable architecture suitable for production workloads* | • *Higher operational costs*<br>• *Increased architectural complexity*<br>• *Vendor lock-in due to reliance on AWS managed services* |

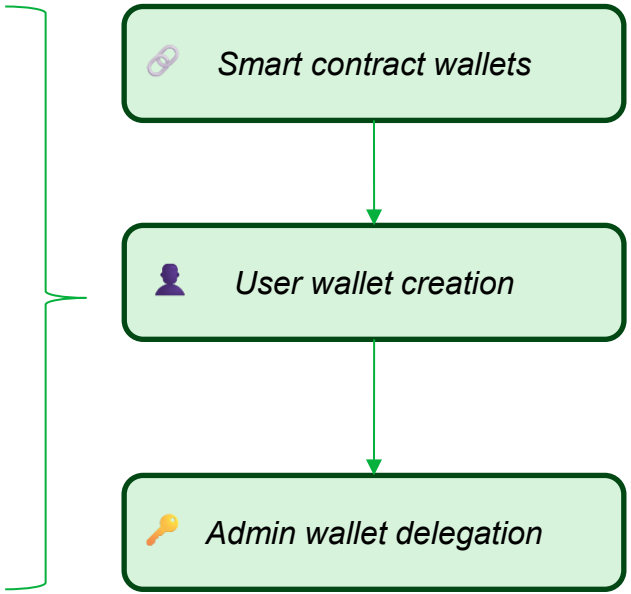
# AGENDA

1. Process overview
2. High-level architecture
3. **Bledger Network Token**
4. Blockchain benefits and transparency
5. Verify data on-chain
6. Bledger screenshots

# ON-CHAIN SMART CONTRACT ARCHITECTURE

Managed Wallet Module (ERC-4337 Account Abstraction)

The **Bledger platform** implements a managed wallet module based on the **ERC-4337** account-abstraction standard. This module enables an advanced, **secure approach to wallet management** with a strong focus on scalability and security – without requiring end users to self-custody their wallets.




Bledger uses open-source **Safe** contracts – the leading on-chain wallet management standard – to instantiate a **multisignature smart-contract wallet** for each user

At registration, the platform deterministically computes a **unique wallet address** for each user

Each wallet is initialized to **delegate an Admin wallet**.

- In the ***PoC*** the key is stored in a configuration file
- In ***Prod*** the Admin wallet's private key can be managed via AWS KMS.

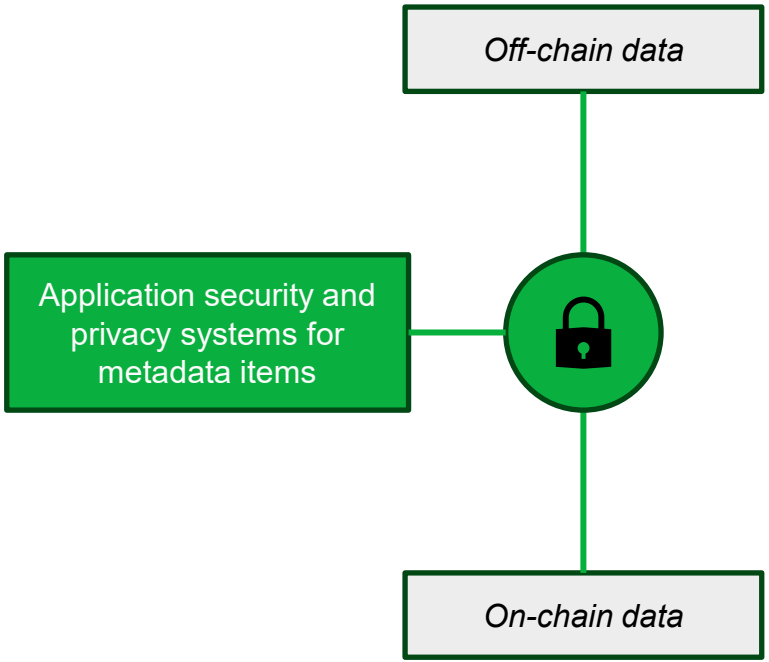
# NETWORK TOKEN

Network Token (ERC721 NFT)


- In BLedger, each Business Network is represented by an **ERC-721 NFT** (smart contract name ***BLedgerNetwork***).
- The token anchor's identity and permissions, while the evolving network state – the tree of companies and documents – lives off-chain as JSON.
- The network is organized as a **hierarchical tree of companies** (parents and children), and its topology can be verified via the Merkle tree root hash stored in the metadata.
- Each token stores a **tokenURI** pointing to a metadata.json and an on-chain **contentHash** to ensure integrity.
- For the **PoC**, the *metadata.json* could be retrieved from an internal endpoint; in the future, storage on IPFS can be integrated.

# PRIVACY

On-chain and off-chain segregation for privacy and security




***Network metadata***

- ***networkId*** – identifier of the network
- ***networkCreated***, – business timestamps for the lifecycle of the network
- ***metadataCreated*** – timestamp of this particular snapshot
- ***merkleHash*** – integrity root over the network tree structure
- ***membersCount*** – number of companies/nodes in the tree
- ***uploadedCount, auditedCount*** – document activity counters
- ***docs*** – array of *DocumentMetadata* entries

***Document metadata***

- ***id*** – unique document identifier
- ***nodeId*** – internal identifier of a company inside the network
- ***documentHash*** – keccak256 document hash

***Network Token***

Each token stores:

- ***tokenURI*** pointing to a metadata.json
- ***contentHash***, an on-chain hash to ensure integrity.

# ON-CHAIN ROLES AND AUTHORITY

Business Network Administrator

- Holds full authority within the system
- The admin can **mint new networks**, control or restrict **transfers**, manage the list of **auditors**, and **update** network-related metadata

Company (owner)

- Represents the entity that **receives the network token**
- The company can **modify** the **metadata** and **manage** the auditors associated with its specific network

Third-party auditors

- Per network–authorized wallets to **update metadata** after verifying network members' documents

# ON-CHAIN | AUDIT TRAIL – 1/2

Bledger blockchain events

The contract emits a set of **Bledger** dedicated **blockchain events** that serve as the authoritative audit log. Indexing these events yields a complete history for each network token:

- **NetworkCreated**(tokenId, to, uri, contentHash) – a network is instantiated and initialized.

- **NetworkUpdated**(tokenId, updater, newUri, contentHash, reason) – the metadata link/hash changes with a typed reason:

- *Generic*: when Business Network administrator changes the tree structure

- *Upload*: when a member of the network uploads his scope files

- *Audit*: when an auditor verifies a network member's scope files

- *RefusedInvitation*: when an invited company refuses to enter the network

Reason can be observed on the blockchain explorer transaction logs (*see next slide*).

- **AuditorSet**(tokenId, auditor, active) – an auditor is added/removed for that network.

# ON-CHAIN | AUDIT TRAIL – 2/2

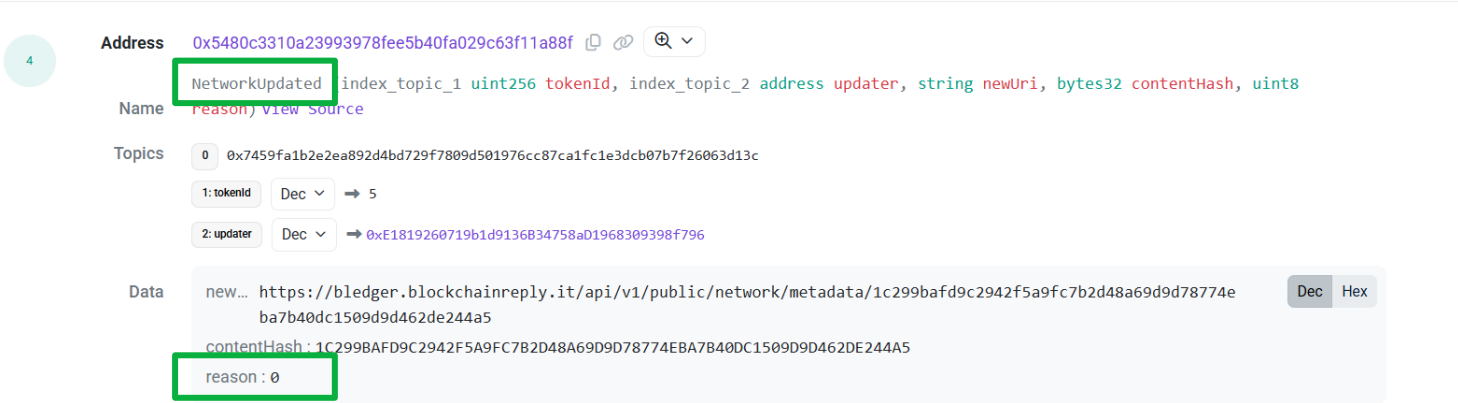
Bledger blockchain events

These events provide a transparent and verifiable audit trail (who changed what, when, and why) and to reconcile off-chain metadata snapshots with the on-chain integrity anchor.

These logs are in addition to standard contract-based logs (such as the ones defined in EIP / ERC standards used).

Every reason has its key:

- Generic: 0
- Upload: 1
- Audit: 2
- RefusedInvitation: 3



# BLEDGER FLOWS