# A blockchain-based Consent Management System for life science data management

Barbara Martelli, Davide Salomoni,
Alessandro Costantini, Luca Giommi,
Ana Velimirović

# The Problem

- On 25th May 2018 a new European Union (EU) law came into effect: the General Data Protection Regulation (GDPR) with the aim of protecting EU individuals' privacy
  - EU individuals = whoever is on EU territory, not only EU citizens
- Many INFN life science research activities involve personal data processing related to healthcare patients and need to comply with the new norm
  - To address this issue, we have built EPIC Cloud (Enhanced Privacy and Compliance Cloud): an ISO 27001 27017 27018 certified partition of INFN Cloud adopting technical and organizational security measures that make it fit for processing personal data
- Among the rights of individuals, there are the rights To Be Informed, To Erasure and To Restrict Processing of personal data
  - We need to add functionalities to enable patients who provide personal data stored in EPIC Cloud to exercise their rights

# The solution
# from an organizational process point of view

- To address the Data Sovereignty issue, the most frequent solution is to ask Data Subjects (in our case patients) to fill and sign a form named “Informed Consent” clearly stating
  - Purposes of data processing
  - Retention period
  - Who is the Data Controller and who is the Data Processor
- Any patient has the right to withdraw his consent at any time
  - In this case Data Controller and Data Processor must delete all his data immediately

# The (partial) solution from a technical point of view

- To manage the Informed Consent workflow and lifecycle, the exploitation of a Consent Management System (CMS) is frequently proposed in literature
  - A CMS should enable data subjects to establish control over their data, giving access permission and audit the use of their personal data, withdrawing permissions and deleting their data
- However, even state-of-the-art CMS still suffer of lack of transparency
  - It is necessary to trust the CMS provider (usually a private company) for the effective deletion of data and compliance to GDPR
  - Trust is often based on the adoption of certification mechanisms (like ISO/IEC 27001), which foresees a third-party independent audit performed on a yearly basis

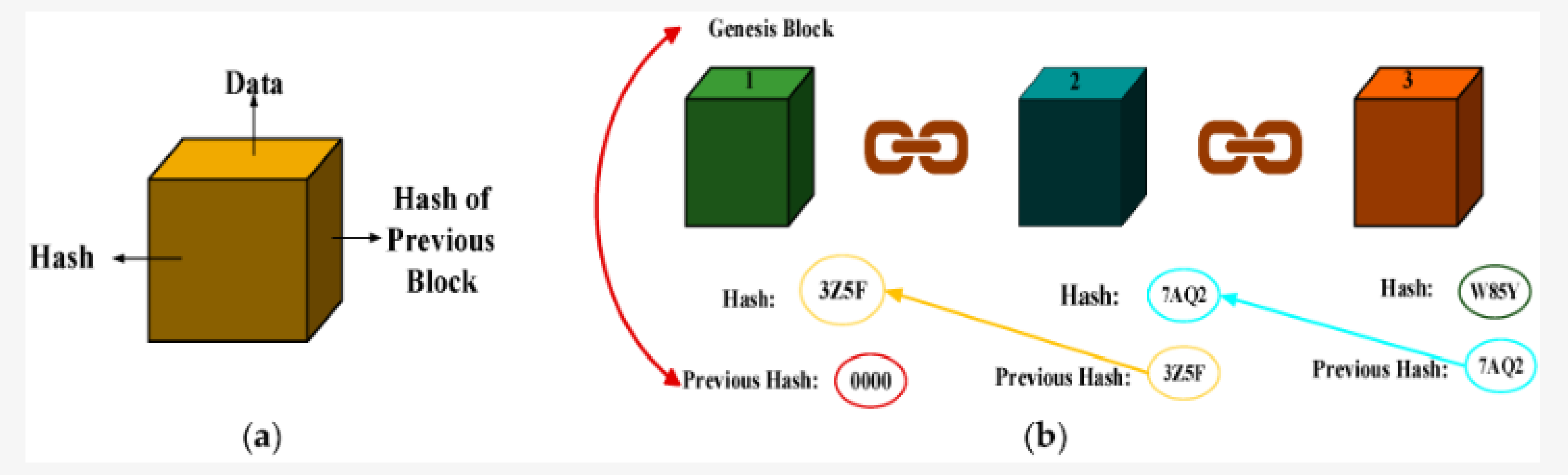
# How to add transparency and trustworthiness to CMSs - The Blockchain

A way of enhancing transparency and trustworthiness to current CMS systems is the exploitation of blockchain technologies

- Distributed ledgers implemented as concatenation of data blocks in a growing chain of immutable elements
- The current value of all ledger values is called the World State (WS)

The chain is maintained by several actors exploiting a combination of

- cryptographic techniques
- consensus algorithms
- peer-to-peer communications
- game theory



# Two main open-source blockchain platforms

### Ethereum

- Access: permissionless
- Deployment: public
- Consensus algorithm: Proof of Stake (PoS)
- Development language: Solidity
- Distributed App (DApp): Smart Contract

### Hyperledger Fabric

- Access: permissioned
- Deployment: private
- Consensus algorithm:  Practical Bizantine Fault Tolerance (PBFT) or dynamic (Sawtooth)
- Development language: golang or JavaScript + Hyperledger Composer
- Distributed App (Dapp): chaincode

### Both

FOSS backed by an international foundation

Ethereum and Hyperledger Fabric
which is the best in our case?

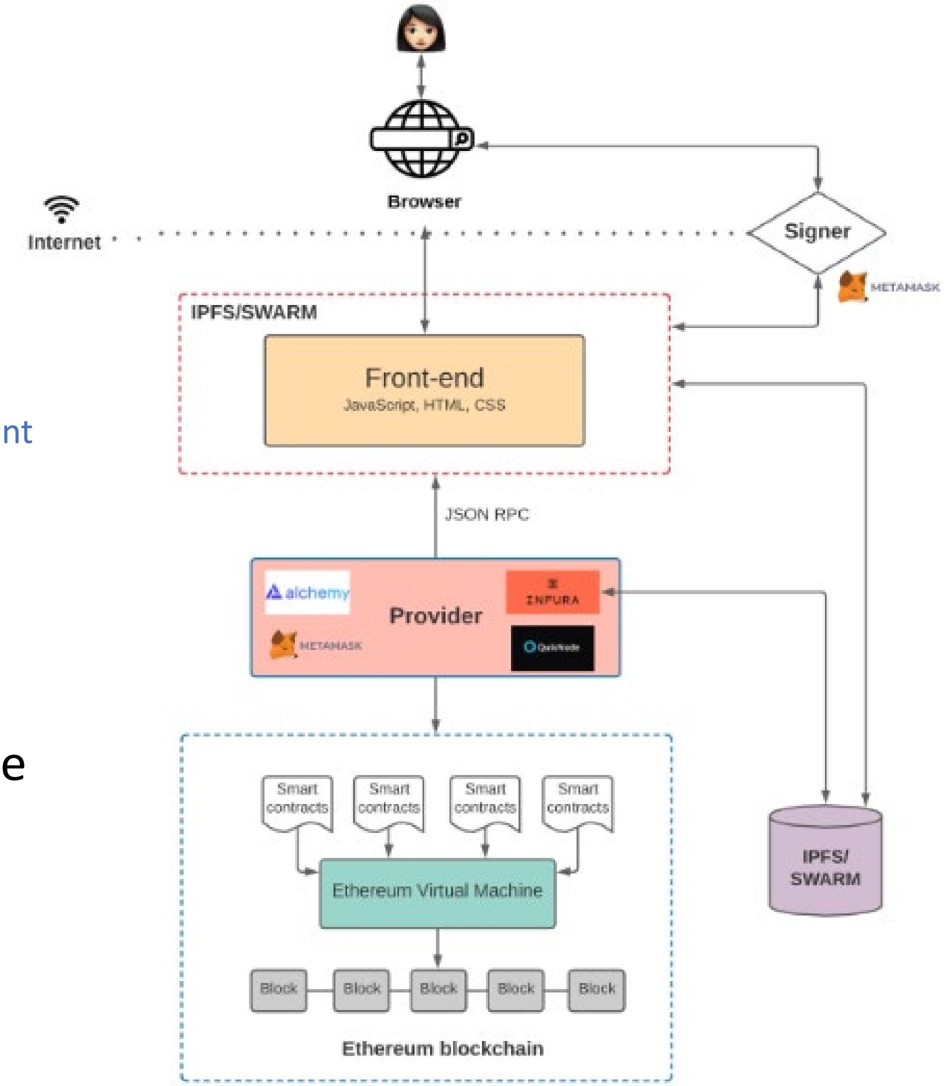
We choose a mix of them:
Hyperledger BESU

# Hyperledger BESU: an Ethereum permissioned blockchain

- While Ethereum architecture (PoS, smart contracts, Solidity) is the most advanced, in our use case the public Ethereum blockchain (mainnet) has some drawbacks:
  - Users can be anonymous, while we need to identify, authenticate and give appropriate roles to all subjects accessing personal data
  - Performance are low and transactions can wait for several minutes to get committed -> eventual consistency
  - Transactions come with a cost: the coin is Ether (ETH) and you need conventional money to get it
- To overcome these limits, we are going to exploit Hyperledger BESU: a permissioned version of Ethereum blockchain that can be deployed in private environments

# What is a DApp

- A DApp (Distributed Application) is an application built on a decentralized network that combines a smart contract and a frontend user interface. It is:
  - **Decentralized** – when DApps operate on Ethereum no one person or group has control, as it is an open public decentralized platform
  - **Deterministic** - DApps perform the same function irrespective of the environment in which they get executed
  - **Turing complete** - DApps can perform any action given the required resources
  - **Isolated** - DApps are executed in a virtual environment known as **Ethereum Virtual Machine** so that if the smart contract has a bug, it won't hamper the normal functioning of the blockchain network
- On Ethereum and Hyperledger BESU DApps are written in the Solidity language
- Solidity is object-oriented, statically-typed and high-level, with syntax influenced by JavaScript and C++



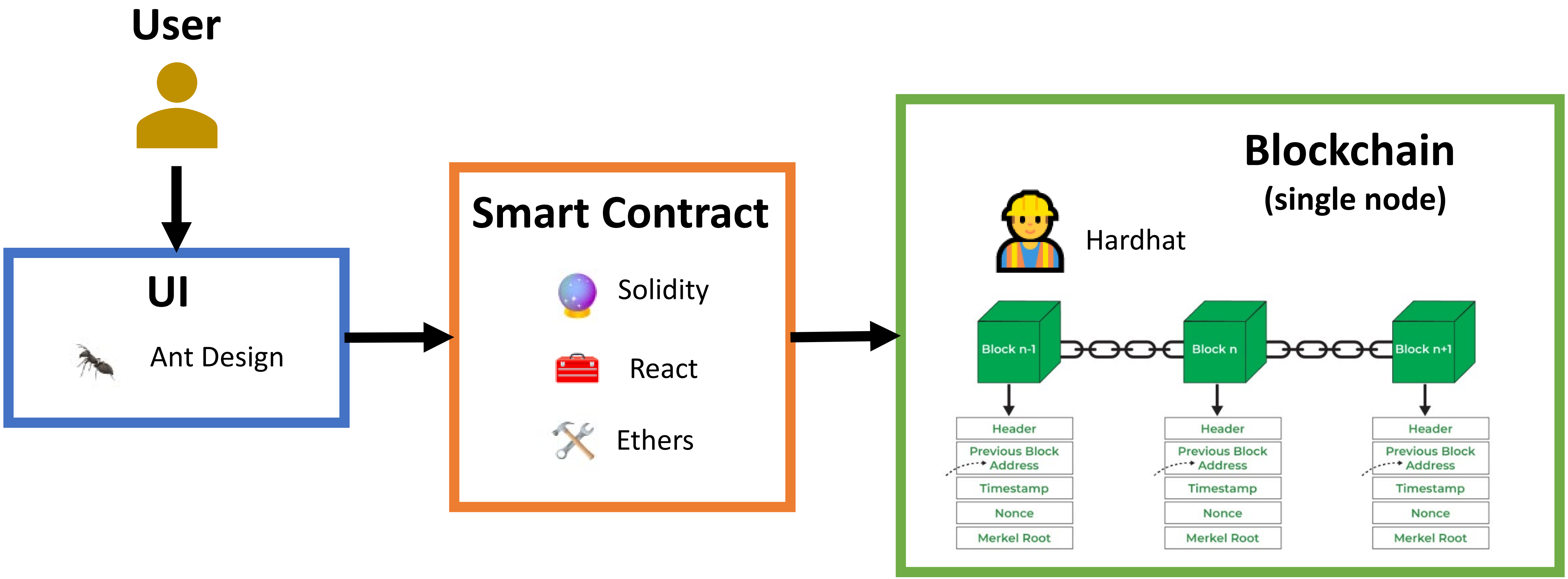
https://www.preethikasireddy.com/post/the-architecture-of-a-web-3-0-application

# What is a Smart Contract

- A smart contract is code that lives on the blockchain. It contains some business logic and a limited amount of data. The business logic is executed:
  - if specific criteria are met by data stored in the blockchain
  - If Participants in the blockchain run the smart contract
- You can think of it as a DApp's backend
  - It's a collection of code (its functions) and data (its state) that resides at a specific address on the blockchain
- Once smart contracts are deployed on the blockchain network, you can't change them

# First step:
# make available a DApp development environment on INFN Cloud

# Ethereum environment for developers: scaffold-eth



User

UI

Ant Design

Smart Contract

Solidity

React

Ethers

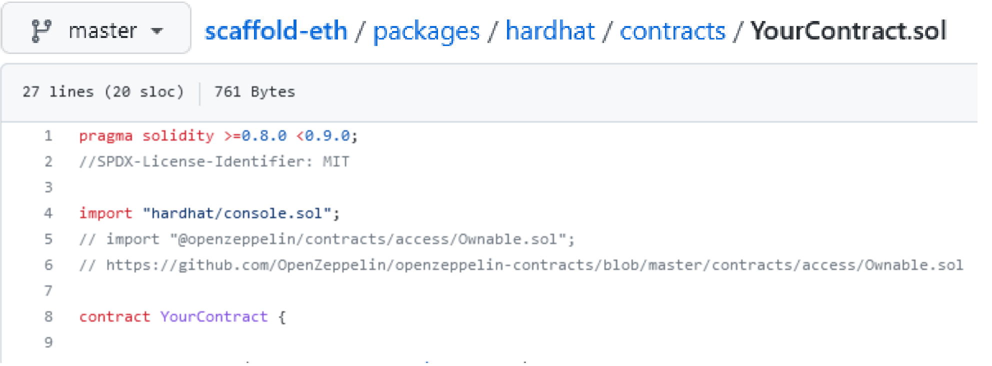
Blockchain

(single node)

Hardhat

# Deploy Scaffold-eth via INFN Cloud

- Scaffold-eth containerization
  - To be portable and reusable
  - Contracts on github repo: every time a contract is created or modified, it will be automatically updated in Scaffold
- Scaffold and Nginx integration
  - Scaffold-eth and Nginx have been integrated in a docker-compose file
  - Nginx is taking care of the correct proxy redirections. It will provide also TLS termination as a further step
- Scaffold-eth deployment customization
  - TOSCA template has been properly customized in order to deploy scaffold-eth application within the INFN Cloud infrastructure

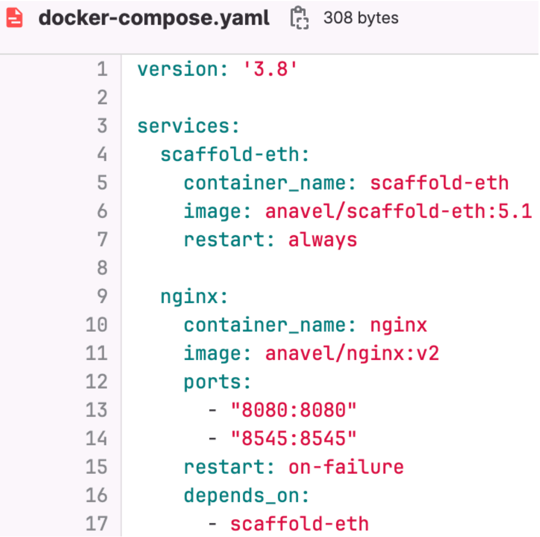


master

scaffold-eth / packages / hardhat / contracts / YourContract.sol

27 lines (20 sloc) | 761 Bytes

```
pragma solidity >=0.8.0 <0.9.0;
//SPDX-License-Identifier: MIT

import "hardhat/console.sol";
// import "@openzeppelin/contracts/access/Ownable.sol";
// https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol

contract YourContract {

```

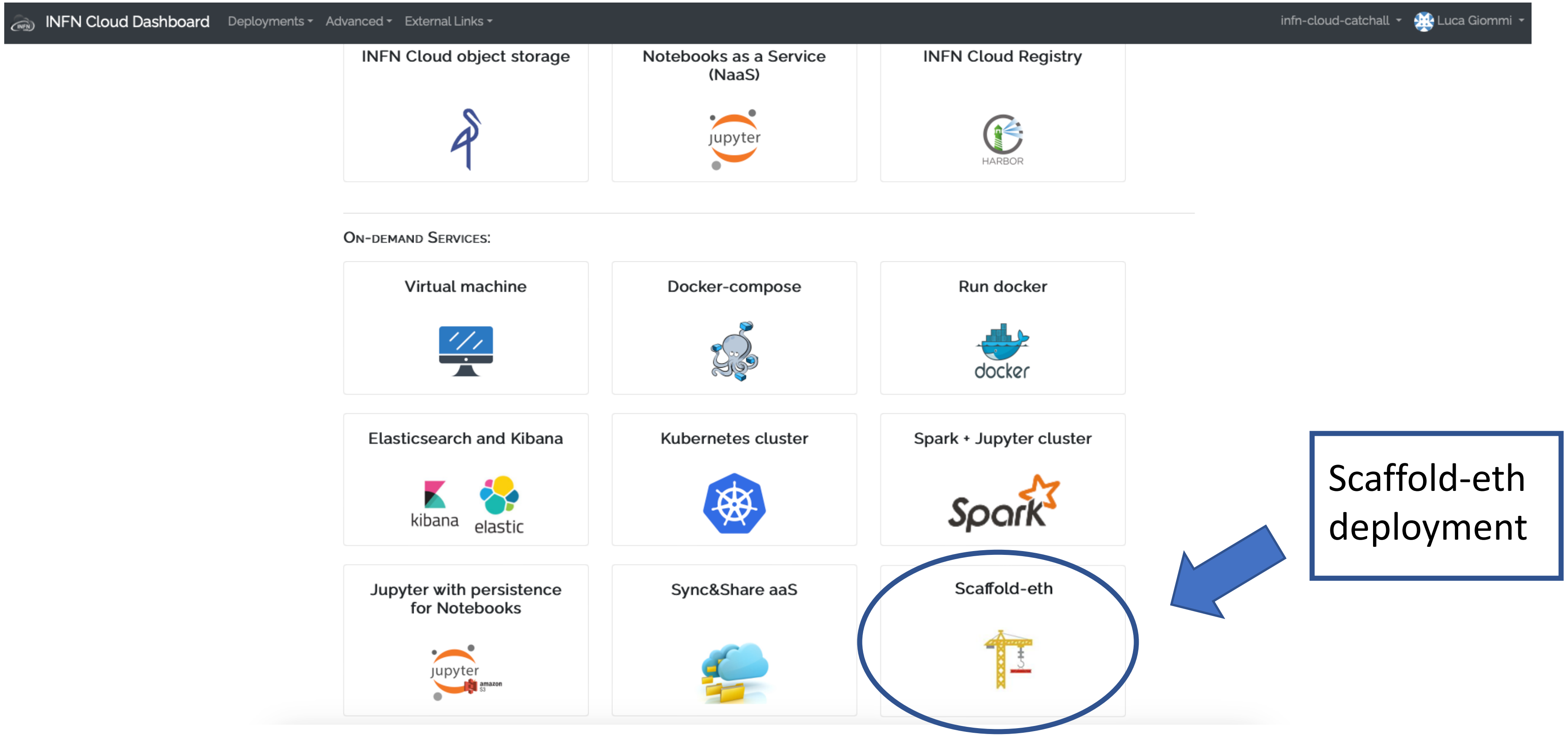


docker-compose.yaml 308 bytes

```
version: '3.8'

services:
  scaffold-eth:
    container_name: scaffold-eth
    image: anavel/scaffold-eth:5.1
    restart: always

  nginx:
    container_name: nginx
    image: anavel/nginx:v2
    ports:
      - "8080:8080"
      - "8545:8545"
    restart: on-failure
    depends_on:
      - scaffold-eth
```

# Scaffold-eth “as a service” tool on INFN Cloud

Scaffold-eth can be deployed on-demand using the INFN Cloud dashboard and related graphical environment



INFN Cloud Dashboard

INFN Cloud object storage

Notebooks as a Service (NaaS)

INFN Cloud Registry

ON-DEMAND SERVICES:

Virtual machine

Docker-compose

Run docker

Elasticsearch and Kibana

Kubernetes cluster

Spark + Jupyter cluster

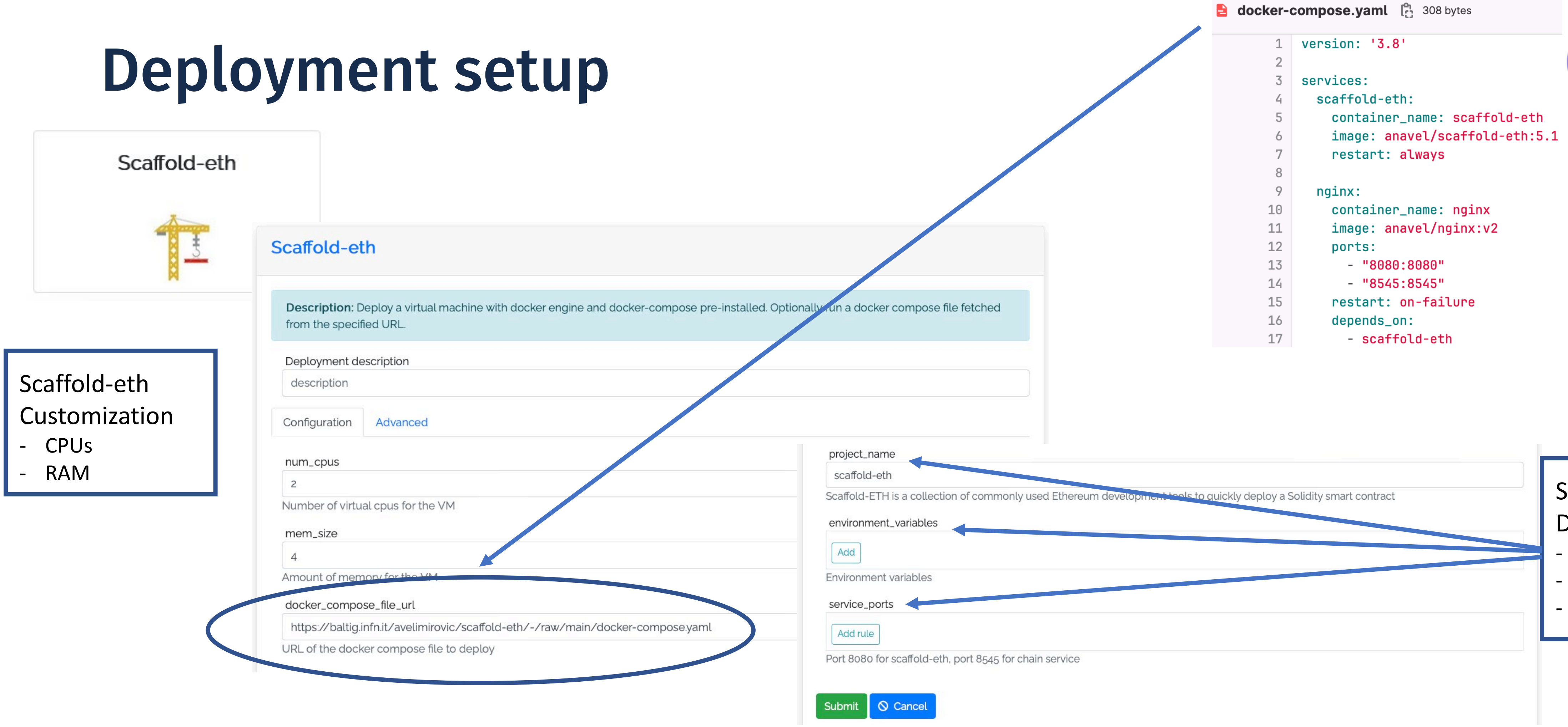
Jupyter with persistence for Notebooks

Sync&Share aaS

Scaffold-eth

Scaffold-eth deployment

# Deployment setup



Scaffold-eth

docker-compose.yaml 308 bytes

```yaml
version: '3.8'

services:
  scaffold-eth:
    container_name: scaffold-eth
    image: anavel/scaffold-eth:5.1
    restart: always

  nginx:
    container_name: nginx
    image: anavel/nginx:v2
    ports:
      - "8080:8080"
      - "8545:8545"
    restart: on-failure
    depends_on:
      - scaffold-eth
```

Scaffold-eth

**Description:** Deploy a virtual machine with docker engine and docker-compose pre-installed. Optionally run a docker compose file fetched from the specified URL.

Deployment description: description

num_cpus: 2
Number of virtual cpus for the VM

mem_size: 4
Amount of memory for the VM

docker_compose_file_url: https://baltig.infn.it/avelimirovic/scaffold-eth/-/raw/main/docker-compose.yaml
URL of the docker compose file to deploy

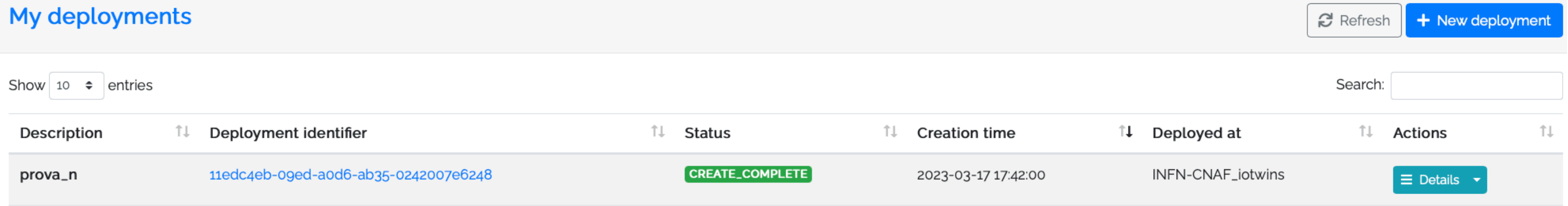
project_name: scaffold-eth
Scaffold-ETH is a collection of commonly used Ethereum development tools to quickly deploy a Solidity smart contract

environment_variables: Add
Environment variables

service_ports: Add rule
Port 8080 for scaffold-eth, port 8545 for chain service

Scaffold-eth Customization
- CPUs
- RAM

Scaffold-eth Default values
- Project Name
- Env variables
- Service ports



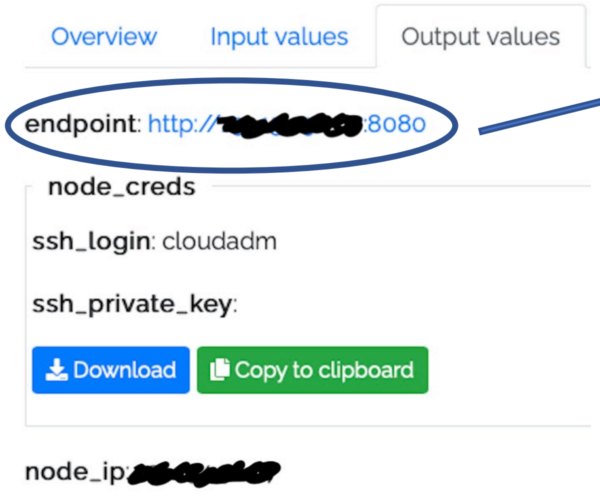
My deployments

Show 10 entries Search:

| Description | Deployment identifier | Status | Creation time | Deployed at | Actions |
|---|---|---|---|---|---|
| prova_n | 11edc4eb-09ed-a0d6-ab35-0242007e6248 | CREATE_COMPLETE | 2023-03-17 17:42:00 | INFN-CNAF_iotwins | Details |

Scaffold-eth Deployment created

# Access Scaffold deployment



Overview | Input values | Output values

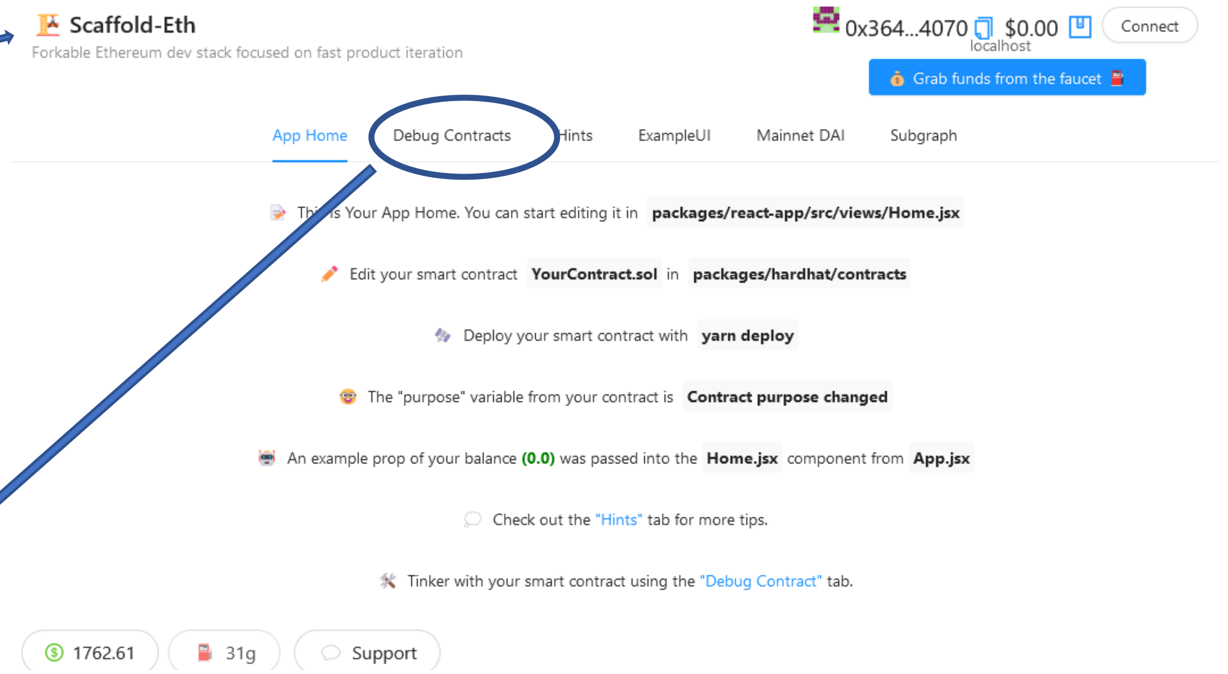
endpoint: http://[redacted]:8080

node_creds

ssh_login: cloudadm

ssh_private_key:

node_ip:[redacted]



This is Your App Home. You can start editing it in packages/react-app/src/views/Home.jsx

Edit your smart contract **YourContract.sol** in **packages/hardhat/contracts**

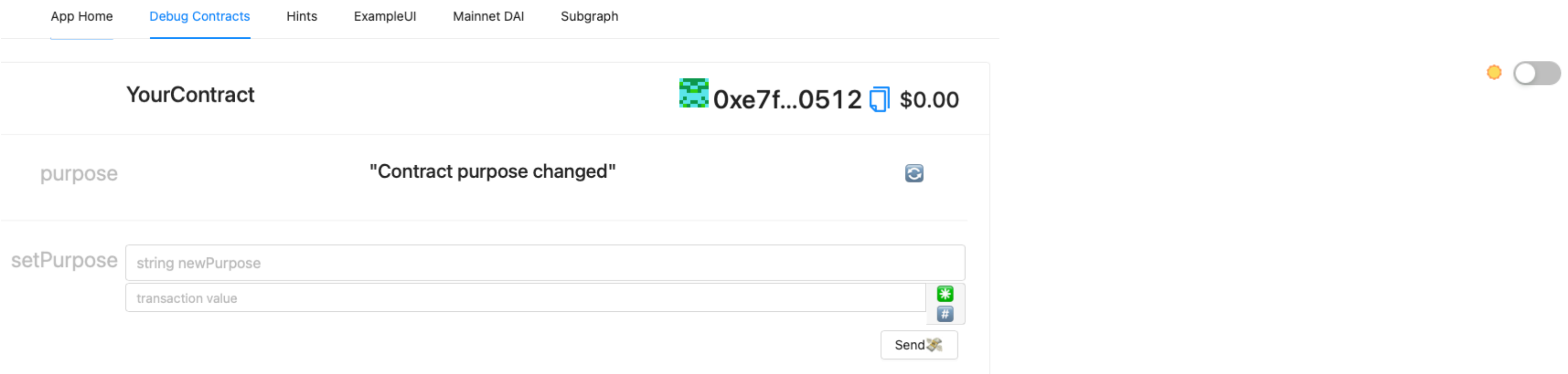
Deploy your smart contract with **yarn deploy**

The "purpose" variable from your contract is **Contract purpose changed**

An example prop of your balance **(0.0)** was passed into the **Home.jsx** component from **App.jsx**

Check out the "Hints" tab for more tips.

Tinker with your smart contract using the "Debug Contract" tab.



YourContract 0xe7f...0512 $0.00

purpose "Contract purpose changed"

setPurpose

# Second step:
# define the CMS architecture and the Informed Consent workflow

# Actors

## Patient

*GDPR Role: Data Subject*

- Give data to Hospital
- Sign the Informed Consent
- Revoke Consent
- Audit the CMS

## Hospital

*GDPR Role: Data Controller*

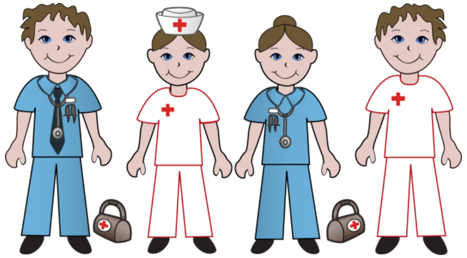
- Collect patients' data
- Appoint INFN Cloud as Data Processor
- Data Consumer: analise Patients' data exploiting the Life Science Datalake

## INFN Cloud

*GDPR Role: Data Processor*

- Get data from hospitals and manage them
- Develop and manage the CMS
- Develop and manage the life science datalake

## Hospital Ethic Committee

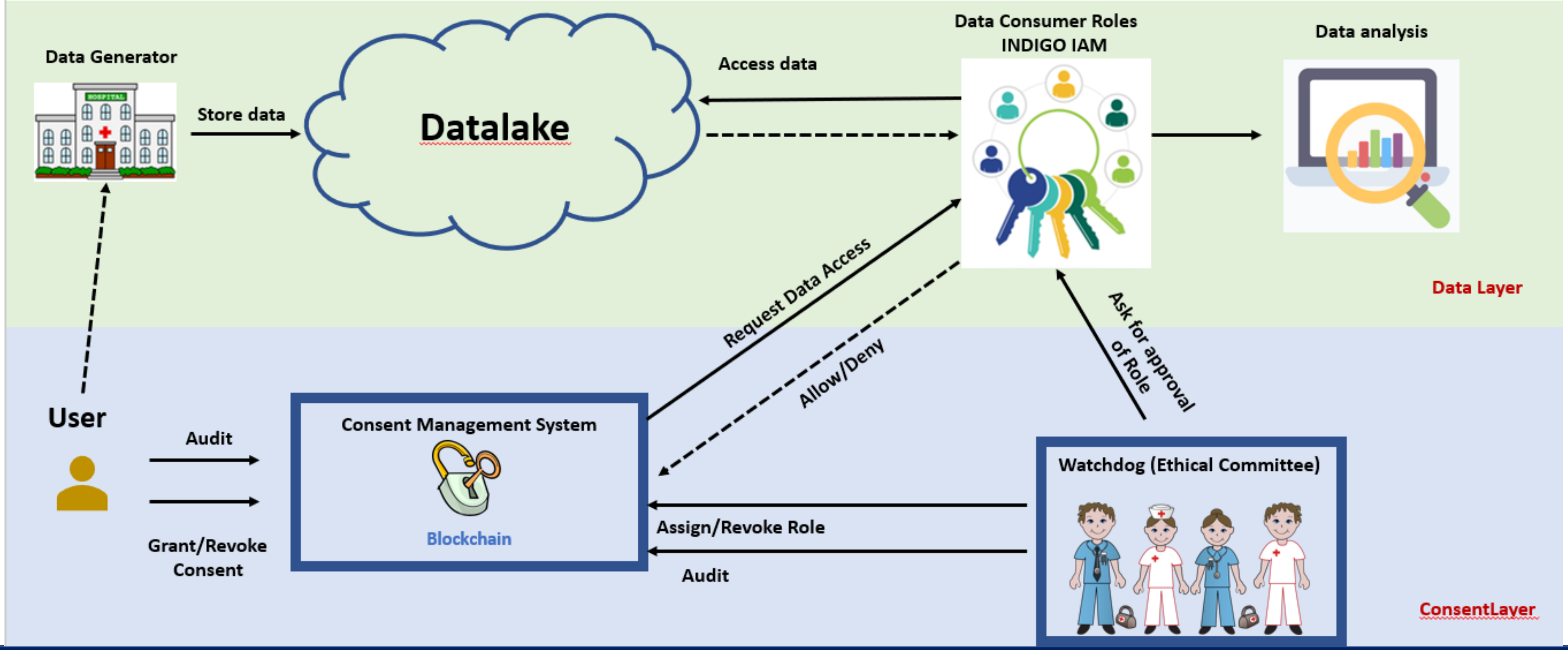


*Role: Whatchdog*

- Assign Roles on CMS and datalake
- Audit CMS

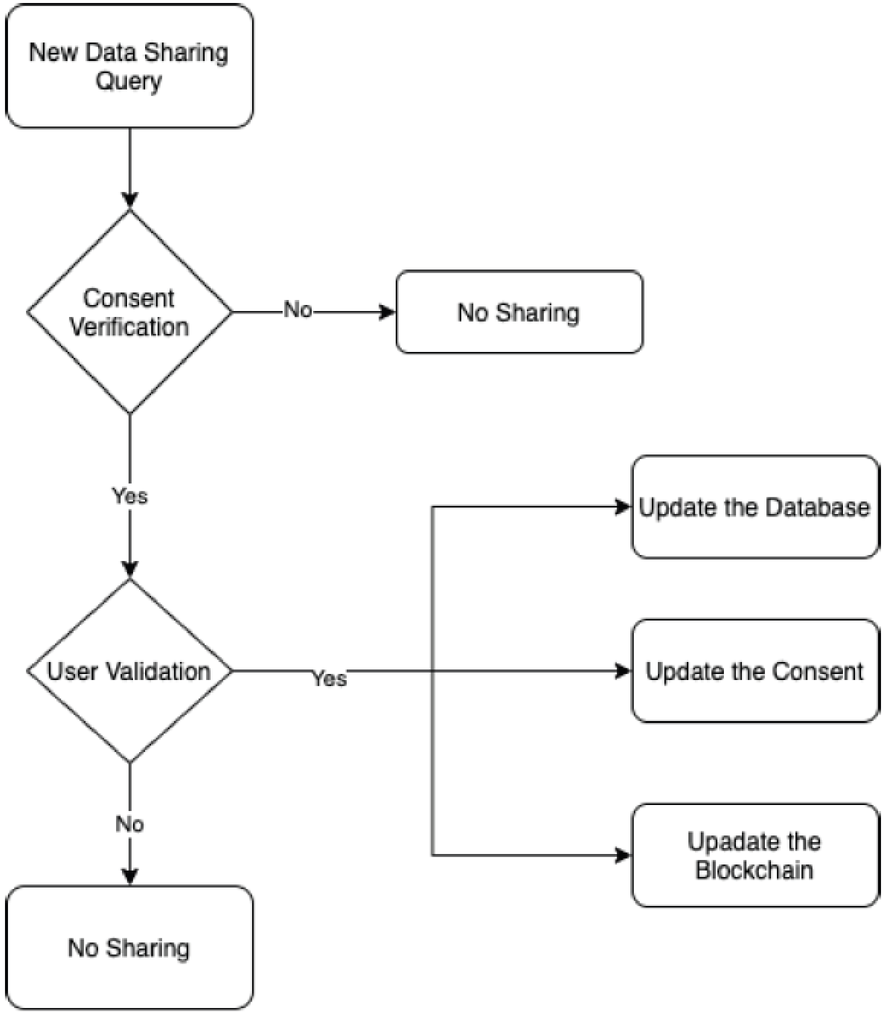
# Consent Management System

- A CMS supports users in controlling who can access their personal data and audit who has accessed their personal data by adding access control and transparency
- Ideally a CMS shouldn't be controlled by any individual/company, that's why often blockchain approaches are considered for its implementation
- A good practice in CMS is to decouple the consent layer from the data management layer
- A CMS must be tamper-proof and auditable



# Informed Consent Workflow

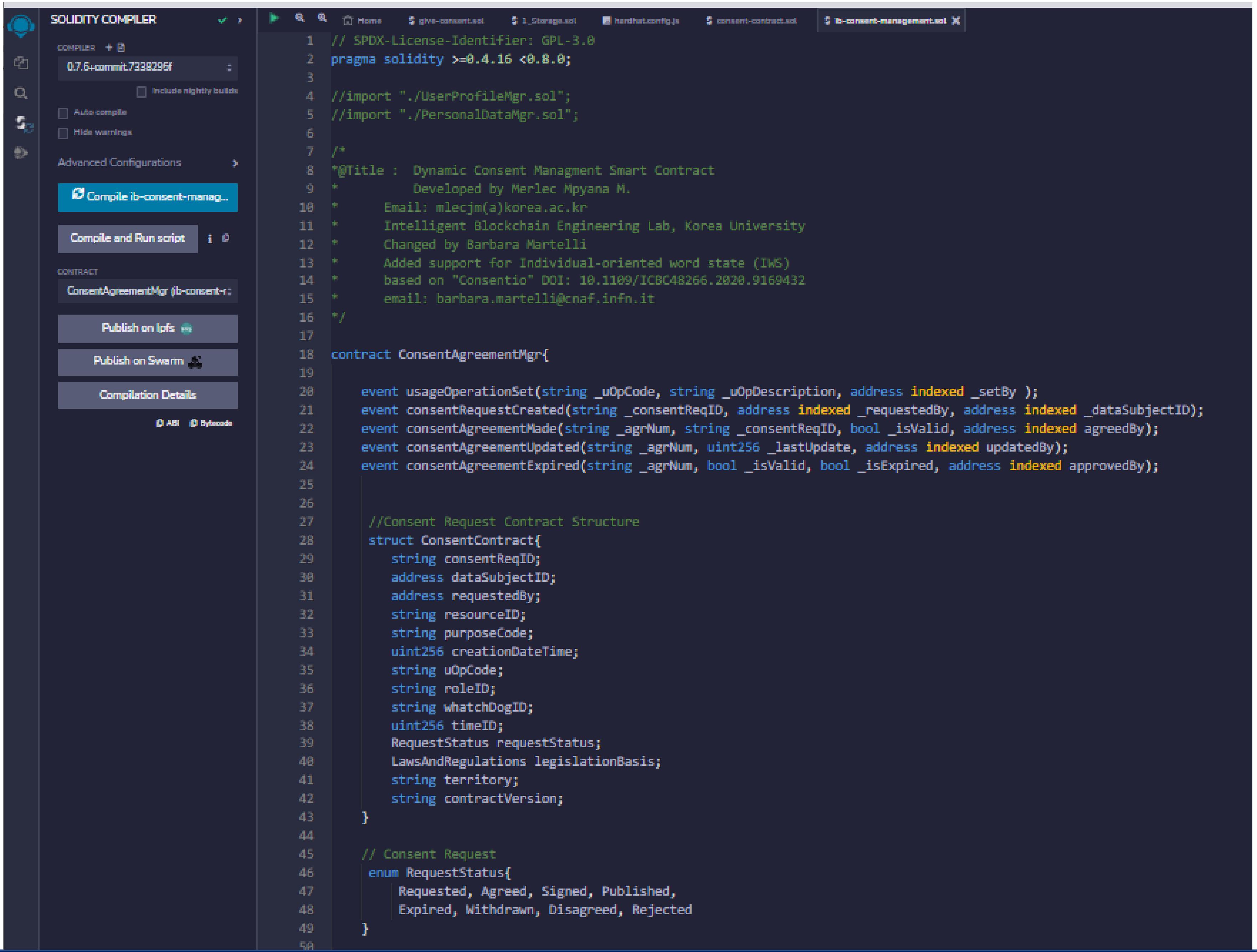
- Patients grant or withdraw consent to data consumers (doctors, researchers, insurance companies, …) acting in particular roles
- Watchdogs (Ethic Committee) assign and revoke data consumers' roles
- Given their roles, data consumers request permission to access data
- The CMS must determine whether such permission can be granted, based on patients consent
- A CMS must be tamper-proof and auditable



# Informed consent smart contracts

The Solidity code for a basic consent management smart contract is here: https://github.com/bmartell/sc-dcms

The code is based on [3]
We added *Individual-oriented word state* (IWS) as suggested by [1]



COMPILER
0.7.6+commit.7338295f

CONTRACT
ConsentAgreementMgr (ib-consent-r:

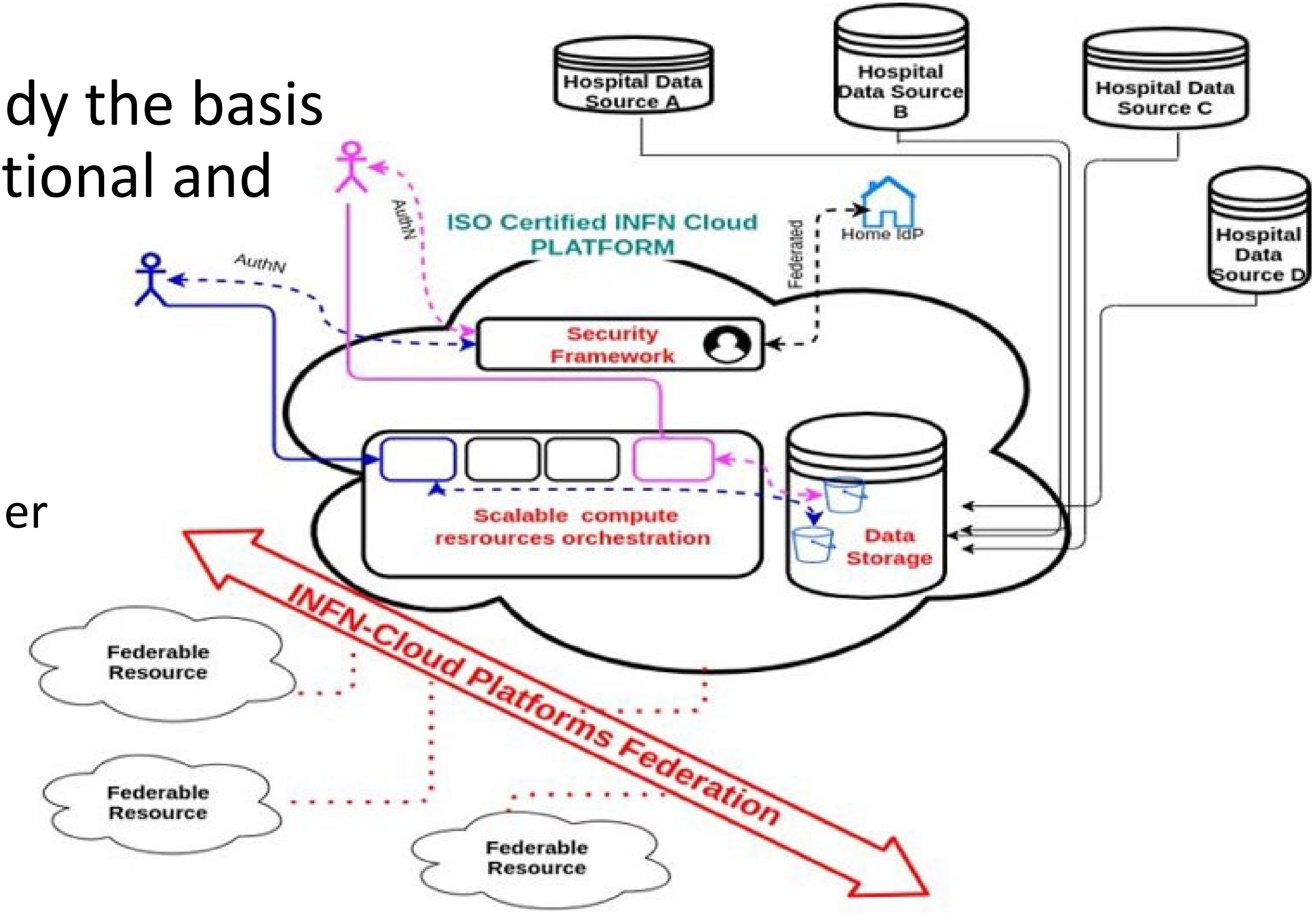
```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.8.0;

//import "./UserProfileMgr.sol";
//import "./PersonalDataMgr.sol";

/*
*@Title :  Dynamic Consent Managment Smart Contract
*           Developed by Merlec Mpyana M.
*       Email: mlecjm(a)korea.ac.kr
*       Intelligent Blockchain Engineering Lab, Korea University
*       Changed by Barbara Martelli
*       Added support for Individual-oriented word state (IWS)
*       based on "Consentio" DOI: 10.1109/ICBC48266.2020.9169432
*       email: barbara.martelli@cnaf.infn.it
*/

contract ConsentAgreementMgr{

    event usageOperationSet(string _uOpCode, string _uOpDescription, address indexed _setBy );
    event consentRequestCreated(string _consentReqID, address indexed _requestedBy, address indexed _dataSubjectID);
    event consentAgreementMade(string _agrNum, string _consentReqID, bool _isValid, address indexed agreedBy);
    event consentAgreementUpdated(string _agrNum, uint256 _lastUpdate, address indexed updatedBy);
    event consentAgreementExpired(string _agrNum, bool _isValid, bool _isExpired, address indexed approvedBy);


     //Consent Request Contract Structure
     struct ConsentContract{
        string consentReqID;
        address dataSubjectID;
        address requestedBy;
        string resourceID;
        string purposeCode;
        uint256 creationDateTime;
        string uOpCode;
        string roleID;
        string whatchDogID;
        uint256 timeID;
        RequestStatus requestStatus;
        LawsAndRegulations legislationBasis;
        string territory;
        string contractVersion;
    }

    // Consent Request
     enum RequestStatus{
         Requested, Agreed, Signed, Published,
         Expired, Withdrawn, Disagreed, Rejected
    }
```

# Conclusions (1/2)
# The Big Picture: toward an open-source genomic datalake for research

INFN Cloud technologies are already the basis for several research projects at national and European level

- Harmony Alliance
- Health Big Data
- Several national projects funded under the EU Recovery and Resilience Plan

We think that blockchain technologies will be of paramount importance to enhance **transparency, auditability** and **trust** at various architectural levels



# Conclusions (2/2)
# INFN Cloud Roadmap to exploit Blockchain technologies in life science applications

1. **BCDEaaS**: on-demand deployment of Blockchain Development Environments based on Ethereum and Scaffold-eth -> Done!

2. **BCaaS**: on-demand deployment of Blockchain general purpose, permissioned blockchain enviroments based on Hyperledger BESU -> work in progress

3. **BC-CMSaaS**: on-demand deployment of blockchain-based Consent Management Systems built on top of the BCaaS functionality -> work in progress

4. **BC-GIMSaaS**: Genomic Information Management System built on top of the BCaaS functionality and exploiting the BC-CMS to manage patient consent -> future work

# Main References

- [1] Rishav Raj Agarwal et al. “Consentio: Managing consent to data access using permissioned blockchains”. In: 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE. 2020, pp. 1–9.
- [2] Darine Ameyed et al. “Blockchain Based Model for Consent Management and Data Transparency Assurance”. 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)
- [3] Merlec MM, Lee YK, Hong SP, In HP. “A Smart Contract-Based Dynamic Consent Management System for Personal Data Usage under GDPR”. Sensors (Basel). 2021 Nov 30;21(23):7994. doi: 10.3390/s21237994. PMID: 34883997; PMCID: PMC8659597
- https://github.com/scaffold-eth/scaffold-eth
- https://docs.scaffoldeth.io/scaffold-eth/
- https://www.cloud.infn.it/