Frederic Christopher Schwill

Towards decentralized and privacy-preserving data marketplaces to unlock data for AI: An examination of Ocean Protocol

Final Master’s Project

Directed by Dr. Josep Domingo Ferrer

Master’s Degree in Computer Security Engineering and Artificial Intelligence

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2021

Content

1. Introduction - 4 -
1.1 Motivation - 4 -
1.2 Goal of the thesis - 4 -
2. Status quo - 5 -
2.1 Data privacy regulation - 5 -
2.2 State of the art - 7 -
2.3 Limitations - 8 -
2.4 Proposal - 8 -
3. Preliminaries - 9 -
3.1 Distributed ledger technology - 9 -
3.2 Smart contracts - 10 -
3.3 Tokens - 10 -
3.4 Standards: ERC20, ERC-721 and ERC-998 - 11 -
3.5 Transactions, keys and addresses - 11 -
3.6 Wallets - 11 -
3.7 Exchanges - 11 -
4. Introducing Ocean Protocol - 12 -
4.1 Data tokens - 12 -
4.2 Data ownership and access - 12 -
4.3 Data transfer - 12 -
4.4 Data wallets - 13 -
4.5 Data exchanges - 13 -
4.6 Data marketplaces - 13 -
4.7 Data privacy - 14 -
4.8 License - 14 -
5. Deep Dive - 15 -
5.1 Architecture - 15 -
5.2 Smart contracts - 16 -
5.2.1 Ocean Token Contract - 16 -
5.2.2 Metadata Contract - 17 -
5.2.3 Data Token Factory Contract - 18 -
5.2.4 Data Token Contract - 19 -
5.2.4 Fixed Rate Exchange Contract - 20 -
5.2.5 Liquidity Pool Factory Contract - 21 -
5.3 Metadata - 22 -
5.4 Pricing - 25 -
5.5 Access Control - 26 -
5.6 Compute-to-data - 27 -

5.7 Protocol flow - 30 -
5.7.1 Publish a dataset - 30 -
5.7.2 Consume a dataset - 33 -
6. Deployment of a privacy-preserving data marketplace prototype - 35 -
6.1 Prototype architecture - 35 -
6.1.1 Infrastructure - 36 -
6.1.2 Data provider - 36 -
6.1.3 Data consumer - 36 -
6.2 Prototype deployment - 37 -
6.2.1 VM1 - Infrastructure - 37 -
6.2.2 VM2 – Data provider - 39 -
6.2.3 VM3 – Data consumer - 39 -
6.3 Prototype demo - 40 -
7. Evaluation - 41 -
7.1 Risks - 43 -
7.1.1 Malicious algorithms - 44 -
7.1.2 Data asset integrity - 45 -
7.1.3 Data token leaks, theft and misuse - 46 -
7.1.4 Model leaks, overfitting, unintended memorization - 47 -
7.1.5 Model poisoning - 48 -
7.1.6 Profiling - 49 -
7.1.7 URL disclosure - 50 -
7.1.8 Smart contract upgradeability - 51 -
7.2 Usability, scalability and costs - 52 -
7.3 Limitations - 53 -
8. Conclusion - 54 -
9. Future research - 56 -
10. Bibliography - 57 -
11. List of Figures - 61 -
12. Appendix - 62 -
12.1 Prerequisites - 62 -
12.2 Ocean provider - 62 -
12.3 Aquarius metadata storage - 63 -
12.4 Ocean.js - 63 -
12.5 React - 64 -
12.6 Compute cluster - 64 -
12.7 Ocean market - 66 -
12.8 Ocean contracts - 66 -
12.8 Barge - 67 -

1. Introduction

1.1 Motivation

While large amounts of valuable data are generated each year, data exchange and analysis are often hindered by restrictions or concerns over security, privacy and trust. Many organizations own data but have no secure and trusted way to share it. Thus, data remains locked. On the other hand, AI researchers, developers and startups regularly aim for large quantities of data, as many developments in AI are dependent on data availability. This especially applies to the training of machine learning models.

A novel possible solution towards privacy-preserving data sharing is Ocean Protocol, an open and decentralized network that aims to connect data providers and consumers securely. Ocean Protocol claims to enable data exchange while preserving privacy and ensuring data ownership, transparency and trust. Ocean Protocol is open-source and permissionless, which appear to be suitable conditions to develop data marketplaces with equivalent properties on top. As of December 2020, Ocean Protocol is still under development with most components being in beta status. A production-ready release is expected to be available in spring 2021. [1]

1.2 Goal of the thesis

This thesis has the goal to examine Ocean Protocol, an open-source toolset for privacy-preserving data exchange, especially concerning its security and privacy properties and resulting risks. First, we describe the state-of-the-art data marketplaces, their limitations and how Ocean Protocol aims to overcome them. We explain its principles and the underlying technology and show, how Ocean Protocol enables privacy-preserving data exchange. The contribution includes the local deployment of a permissionless data marketplace prototype on Ocean Protocol[1], which showcases a solution enabling global privacy-preserving data sharing across organizations to be used to train machine learning models. Moreover, the thesis assesses, if and under which circumstances Ocean Protocol can be used in practice to build open and privacy-preserving data marketplaces on top. The analysis primarily evaluates the resulting risks from a business perspective, with respect to corporate risk management and including relevant privacy regulation. Finally, it elaborates on how to mitigate the risks to an acceptable level and advises on how to securely share and exchange data on data marketplaces built on Ocean Protocol.

[1] Ocean Protocol is developed by BigchainDB GmbH [2] under the oversight of Ocean Protocol Foundation (OPF) [3]. All tools are open-source and available under Apache 2.0 license on Github [4]. The license allows to freely use, modify and distribute the software as long as modifications and distributions contain a copy of the license, a list of modifications and a reference to the copyright owner. Ocean Protocol software used in this work was cloned or forked and modified from Ocean Protocols Github repository [4].

2. Status quo

2.1 Data privacy regulation

Any data sharing solution, such as data marketplaces on Ocean Protocol, need to comply with privacy regulations as soon as personal data is involved. The relevant EU law is the 'General Data Protection Regulation' (GDPR), aiming to give individuals control over their personal data. The GDPR applies to any enterprise processing personal data of individuals inside the European Economic Area (EEA).

Personal data "means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person." [GDPR, Article 4.1] The regulation "applies to the processing of personal data wholly or partly by automated means and to the processing other than by automated means of personal data which form part of a filing system or are intended to form part of a filing system." [GDPR, Article 2.1]

Following that, a privacy-preserving data marketplace that involves personally identifiable information (PII) needs to guarantee GDPR-compliance if it aims to be adopted and used by EU enterprises. Essential compliance requirements are

- to collect data "for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes; further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall, in accordance with Article 89(1), not be considered to be incompatible with the initial purposes ('purpose limitation')" [GDPR, Article 5.1b]

- to process personal data "in a manner that ensures appropriate security of the personal data, including protection against unauthorized or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organizational measures ('integrity and confidentiality')". [GDPR, Article 5.1f]

- to implement "appropriate technical and organizational measures, such as pseudonymization, which are designed to implement data-protection principles, such as data minimization, in an effective manner and to integrate the necessary safeguards into the processing in order to meet the requirements of this Regulation and protect the rights of data subjects."
  [GDPR, Article 25.1]

- to implement "appropriate technical and organizational measures to ensure a level of security appropriate to the risk, including inter alia as appropriate:
  - a) the pseudonymization and encryption of personal data;
  - b) the ability to ensure the ongoing confidentiality, integrity, availability and resilience of processing systems and services; […]

  In assessing the appropriate level of security account shall be taken in particular of the risks that are presented by processing, in particular from accidental or unlawful destruction, loss, alteration, unauthorized disclosure of, or access to personal data transmitted, stored or otherwise processed." [GDPR, Article 32]

The data owner, which is referred to as ‘data controller’ in the GDPR, is responsible to “implement appropriate technical and organizational measures to ensure and to be able to demonstrate that processing is performed in accordance with this Regulation.” [GDPR, Article 24.1] Infringements are “subject to administrative fines up to 20,000,000 EUR, or in the case of an undertaking, up to 4 % of the total worldwide annual turnover of the preceding financial year, whichever is higher” [GDPR, Article 83.5]. Moreover, “any person who has suffered material or non-material damage as a result of an infringement of this Regulation shall have the right to receive compensation from the controller or processor for the damage suffered.” [GDPR, Article 82.1]

To conclude, a privacy-preserving data marketplace to be used by EU enterprises needs to protect data subjects' rights by fulfilling the principles of security and privacy by design and by default. It needs to implement suitable technical and organizational measures appropriate to the risk to ensure the ongoing data confidentiality, integrity and availability.

Responsibility is with the data controller, who needs to be able to demonstrate that all data processing is performed in accordance with GDPR. Depending on the severity, violations can be fined with up to 20,000,000 EUR or up to 4 % of the total worldwide annual turnover, whichever is higher. Any person who has suffered damage due to such a violation has the right to receive compensation.

Given the strict EU regulations and the resulting financial risks regarding PII data breaches, a privacy-preserving data marketplace needs to follow the ‘security by design’ principle and thoroughly address all GDPR requirements.

2.2 State of the art

Over the past years, a lot of data marketplaces and exchanges appeared that allow for global data sharing and trade. However, as GDPR prohibits corporations from sharing data containing personally identifiable information (PII), affected data assets cannot be shared, unless they were anonymized[2] or pseudonymized[3] beforehand. However, proper anonymization of large data assets is a complex and costly task. Another challenge with many traditional data marketplaces is that data owners lose control once a data asset is sold and transferred to the buyer. In many cases, the buyer can potentially distribute or resell the data. Because of that, many organizations refrain from sharing their data, and the data remains locked.

These circumstances led to the development of privacy-preserving data marketplaces, that represent the state of the art today. Examples are "Data Republic" [6], "OPAL" [7] and "X-Road" [8]. They all have in common that an algorithm is moved to the data. The data never leaves its repository to comply with GDPR and only aggregated, and thus anonymized, answers are returned to the buyer. Algorithms are open so that their trustfulness can be inspected. Corporations can sell compute access on their data assets, while they do not sell the data itself. The method has the advantage to allow for data monetization while ensuring GDPR compliance and data ownership. The following figure shows the architecture of the "Data Republic" marketplace.

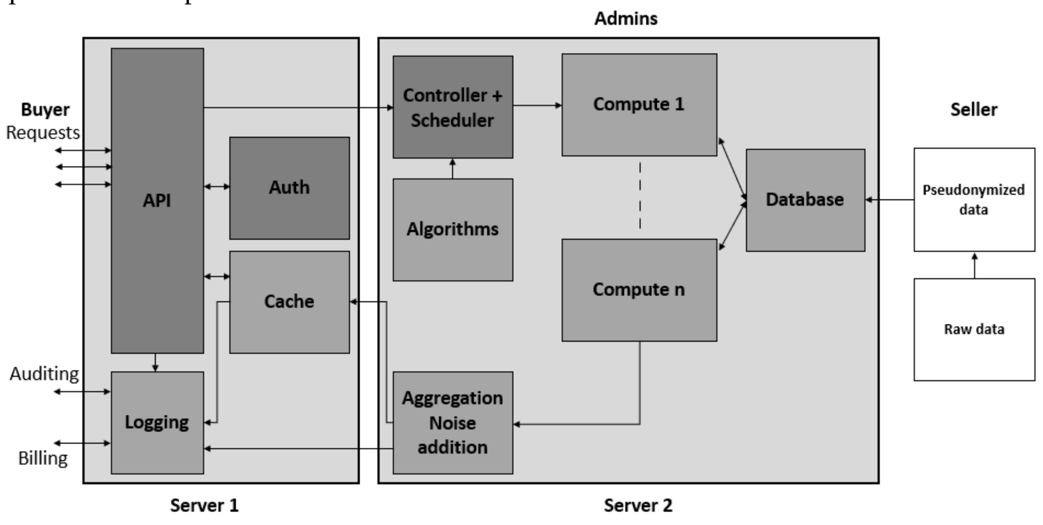


Figure 1: Data Republic Architecture

When applied to machine learning, the approach of selling remote computation on private data assets is similar to federated learning. With federated machine learning, a central server holds a global machine learning model. A copy of the model is sent from the server to the place where the data is stored and trained locally. Afterwards, the local model is sent back and aggregated on the central server, refining the global model, without exposing any of the private data it was trained on. Federated learning eliminates the need to move the data to a central repository for the purpose of training. Applied to privacy-preserving data marketplaces, as the computation is brought to the data, buyers can train a prediction model across many data silos, while the data is kept in its respective repository. Provided the algorithm is trusted and does not leak any PII, models can be trained on PII while assuring GDPR compliance. State-of-the-art and most prominent federated learning projects are TensorFlow Federated [9] and OpenMined [10]. However, both are missing marketplace functionality.

[2] Anonymization is a "process by which personal data is irreversibly altered in such a way that a data subject can no longer be identified directly or indirectly, either by the data controller alone or in collaboration with any other party" [5].

[3] Pseudonymization is a process that replaces fields containing PII with artificial identifiers or pseudonyms. The difference to anonymization is, that pseudonymized data can be restored to its original state, while anonymized data can never be restored.

2.3 Limitations

The privacy-preserving data marketplaces that represent the state of the art today, like "Data Republic", "OPAL" and "X-Road" allow for GDPR compliant data sharing. An algorithm is moved to the data, and the data never leases its repository. Only aggregated, and thus anonymized, answers are returned to the buyer. The technology of said marketplaces is an improvement compared to the traditional marketplace approach that relies on downloads and thus lacks GDPR compliance. However, they introduce new drawbacks and associated risks to corporations:

- Data owners lose control over their data.
- The data needs to be deposited in custodial repositories managed by a third party. The repository (called "Data Sandbox" within Data Republic, "Database" within OPAL, "Security Server" within X-Road) is in control of the marketplace operator.
- Marketplaces are managed by a centralized entity, a valuable target for attackers.
- Marketplace/repository operators have full control over customers, data providers and data.
- Data needs to be pseudonymized before depositing into the repository, which is costly.
- The deposit of non-anonymized/pseudonymized data into the repository is questionable, as it contradicts the GDPR principle of purpose limitation. It states that "data shall be collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes; further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall, in accordance with Article 89(1), not be considered to be incompatible with the initial purposes." [GDPR, Article 5.1b]
- The security of the solution is mainly dependent on the operator of the data repository.

2.4 Proposal

A novel possible solution is Ocean Protocol, which aims to overcome the limitations of current privacy-preserving data sharing technology. Ocean Protocol is a set of tools that combine data marketplaces, the idea of federated machine learning and distributed ledger technology to form an open and decentralized network based on blockchain that connects data providers (data sellers) and consumers (data buyers). It claims to enable data exchange while preserving privacy and ensuring data ownership, transparency and trust. Ocean Protocol eliminates many drawbacks of the state-of-the-art data marketplaces by design:

- There does not exist a central entity. The network is based on blockchain and is community-governed.
- Data owners keep full control over their data.
- The data stays on-premise. Data never leaves its location, as there is no external repository.
- It follows the principle of purpose-limitation because the data is never moved anywhere.
- The security of the solution is mainly dependent on the distributed ledger and the data owner.
- It follows a permissionless approach, allowing for data sovereignty of organizations and individuals.
- The underlying blockchain allows for auditability, transparency and trust.
- The underlying blockchain offers censorship-resistance.

3. Preliminaries

Ocean Protocol is primarily based on distributed ledger technology (DLT). The following chapter gives a brief introduction to the foundation of DLT. It introduces the standards and tools necessary to build a data-sharing solution that satisfies the requirements listed above.

3.1 Distributed ledger technology

"Distributed ledger technology (DLT) allows the maintenance of a global and append-only data structure by a set of mutually untrusted participants in a distributed environment." [11] It holds a consensus of replicated, shared and synchronized data, geographically distributed across the globe, based on a peer-to-peer network. There exists no central authority or administration. Given these features, distributed ledgers offer immutability, accountability and censorship resistance. The characteristics of a distributed ledger make it suitable to run a distributed payment system on top, with Bitcoin [12] being the most known representative today.

While Bitcoin is based on a blockchain, the shaping of such a ledger is not limited and could be a graph, lattice or similar. Moreover, the transaction does not necessarily have to be grouped into blocks. To build a data-sharing solution on top, most types of distributed ledgers might be suitable. Nevertheless, for simplicity and interoperability, we will keep the focus on blockchains.

Blockchain

As mentioned before, the blockchain is a subset of distributed ledger designs. It is characterized by its structure of blocks that form an immutable chain. Blocks hold batches of transactions. All blocks are linked by using a cryptographic hash function, and every block contains a cryptographic hash of the previous block. The chain of hashed blocks consequently guarantees immutability because altering a block is not possible without altering all following blocks. As the chain is geographically replicated across all participating nodes, an attacker needs to control the majority of those nodes which is infeasible in practice. Furthermore, because of the geographical replication, blockchains can be described as secure and highly available by design.

Blockchains can be permissioned or permissionless. Permissionless blockchains have no access control and thus are open and public so that anyone can add nodes, transactions and applications to the network without requiring the trust or approval of others. On the other hand, permissioned blockchains contain an access control layer that governs access to the network. They are often referred to as 'consortia blockchains'.

For the purpose of building a decentralized and open data sharing solution, a permissionless ledger appears to be most suitable. The core functionality of such a solution is an escrow function that manages payment and data delivery in case pre-defined conditions are met. The functionality can be implemented on top of a blockchain by using smart contracts.

3.2 Smart contracts

A computer program that automatically executes, controls or documents events according to the terms of a digital contract is referred to as a smart contract. [13] It is a collection of code (functions) and data (state) that resides on the blockchain [14]. Specifically, the code is compiled into bytecode, and its content and state are written into a blockchain transaction. The program can be triggered by blockchain transactions and is then executed on every node of the network. Consequently, all nodes hold the state changes corresponding to the replicated execution of the program. Given that the blockchain features are inherited by the smart contract layer, nobody has authority over the program execution. Smart contracts can be used to implement tokens, ownership, voting and similar logic on the blockchain. They guarantee and allow for immutability, transparency and auditability in a permissionless and trustless environment.

The most common and popular smart contract implementation today is based on the Ethereum blockchain [15]. Ethereum offers a Turing-complete programming language and a smart contract framework on its blockchain, that became a de-facto standard within the permissionless blockchains. The corresponding programming language is called Solidity, which is an object-oriented language to develop smart contracts that run on the Ethereum nodes, isolated in a runtime environment called Ethereum Virtual Machine (EVM) [16]. It can be used to implement applications on top of the Ethereum blockchain that enforce a pre-defined logic and offer a non-repudiable record of transactions [17]. Every transaction and computation within the Ethereum network costs fees, which are called ‘gas fees’.

It must be noted that the replicated execution of smart contracts is costly in terms of computation and gas fees and therefore only appropriate for reasonably simple programs. For the purpose of decentralized data exchange, a basic escrow functionality is needed, that grants access do a data set if a condition is fulfilled. If the data consumer issued payment, then the smart contract should grant access to the data and forward the payment to the data seller. The logic appears to be reasonably simple to be implemented in a smart contract.

To permit the smart contract to interact with an asset like a dataset, a representation of said asset needs to be present on the blockchain. This can be realized by tokenizing the asset.

3.3 Tokens

In general, a token represents a digital asset. The range of represented assets is broad; it can be anything from real-world or digital objects to units of a currency. Tokens can be fungible (FT) or non-fungible (NFT), which describes the property of interchangeability. Given a fungible token, it is interchangeable because every token of a kind has the same type and value. These tokens are suitable to represent, for example, a unit of a currency or a share in a company. Fungible tokens in Ethereum are based on the ERC-20 standard (Ethereum Request for Comments 20) [18].

In contrast, a non-fungible token is unique and can have a different value than another token from the same contract. It can represent a unique asset like a lottery ticket, a physical property, an access key or negative valued assets like loans and responsibilities. Non-fungible tokens in Ethereum are based on the ERC-721 standard [19].

3.4 Standards: ERC20, ERC-721 and ERC-998

The ERC-20 and ERC-721 standards introduce a specification for fungible/non-fungible tokens and define an API within Smart Contracts so that token-related conditions and actions like a token transfer can be implemented into a contract. Hence, an ERC-20/ERC-721 token itself is deployed as a smart contract on the Ethereum blockchain. Moreover, the standard introduces a defined set of commands to interact with software tools and applications like wallets, exchanges, marketplaces and similar.

Because it only takes a smart contract to deploy an ERC-20/ERC-721 token to the blockchain, tokens are simple and easy to deploy at scale.

3.5 Transactions, keys and addresses

A transaction is a cryptographically signed instruction issued from an account. In its simplest form, a transaction transfers a token from one account to another.

To be able to issue transactions and to send and receive tokens, each account needs to generate a public/private key pair. The private key is utilized to send transactions to an address of another participant. Addresses are used to receive transactions and are derived from the public key. Precisely, in Ethereum keys are based on the Elliptic Curve Digital Signature Algorithm (ECDSA) [20], and an address is composed with the prefix “0x” concatenated with the rightmost 20 bytes of the public keys Keccak-256 hash. Hence, to send an Ethereum transaction, one needs to know the private key of the spending address and needs to calculate the Keccak-256 hash of the receivers’ public key.

3.6 Wallets

A wallet stores the private and public key of a participant. It is a software tool used to store and manage the keys to send and receive transactions. In general, a wallet is not necessary to participate in a blockchain network but increases usability and convenience and is useful when managing transactions and tokens.

3.7 Exchanges

Exchanges offer the possibility to exchange tokens. Because tokens represent an asset, token exchanges can be used to exchange any asset represented by a token. Exchanges in a traditional sense can be centralized (CEX), where a trusted instance oversees the process of exchanging assets but also decentralized (DEX) based on smart contracts.

4. Introducing Ocean Protocol

To enable a decentralized data sharing solution, Ocean Protocol takes the blockchain standards, and tools described in the preceding chapter and apply them to data assets. In general, a data asset can be any dataset or any algorithm stored anywhere. The following chapter describes the general concepts behind Ocean Protocol:

- Tokens become data tokens
- Private keys become data access keys
- Token transfers become data (access) transfers
- Wallets become data wallets
- Exchanges become data exchanges
- Data marketplaces allow to buy and sell data services

4.1 Data tokens

Tokens become data tokens. A data token represents the right to access a data asset. Hence, each data asset gets its own token. [21] Data tokens can be (fungible) ERC-20 tokens which means that each token grants equal rights to access the data. This enables it to be accessed multiple times. If individual access control is needed, data tokens can be ERC-721 which makes them non-fungible.

The purpose of a data token is similar to traditional access/authorization tokens, such as OAuth 2.0. OAuth 2.0 uses tokens to delegate and authorize third-party access to server resources on behalf of the owner without sharing credentials [22]. Using OAuth in practice, a user can grant access to web services and applications without giving away his password. The mental model is comparable to Ocean Protocol data tokens, where the tokens are used to grant access to a data service. The main difference is that traditional access tokens such as OAuth tokens are technically a string of characters that can be copied and transferred limitless, potentially granting access to everyone that obtains a copy of the token. On the other hand, an ERC token can only be spent once. The double-spend problem is of no concern because token spends (transfers) are processed on the blockchain, which prevents double spends by design.

4.2 Data ownership and access

Ownership of a data token is equivalent to ownership of the represented data set. To own a data token and the corresponding data set, one needs to hold the private key to the token. To access a data set, the data consumer needs to hold one data token. Access to a dataset is managed by a smart contract that grants access as soon as it receives one data token from a consumer. A license is attached to the data token that specifies the terms of use and copyright to comply with intellectual property regulation.

Access properties are essential for privacy-preserving data exchange, which will be later discussed in detail.

4.3 Data transfer

Ownership of a token implies the right to transfer the token (unless specified otherwise in the license attached). A transfer of a data token transfers the access right to the data. Because data tokens are built on the ERC-20/ERC-721 standards, they are integrated within the Ethereum ecosystem, and they can be transferred with any compatible software or device. Following that, with Ocean Protocol, it becomes possible to transfer data access rights on the blockchain.

4.4 Data wallets

As data tokens are ERC-20 or ERC-721 tokens, it becomes feasible to leverage existing blockchain technology to manage and store data access rights. Ocean Protocol enables it to use regular Ethereum-compatible blockchain wallets for data token custody and management. These can be software wallets (mobile or desktop such as Metamask [23], MyEtherWallet [24] and others), hardware wallets (like Trezor [25] or Ledger [26]), paper wallets (write down the private key) or brain wallets (memorize the key). Using a hardware wallet is probably a suitable solution for enterprises, as the private key is stored inside the wallet and never leaves it because the hardware wallet itself signs transactions. This can be further secured by using multi-sig hardware wallets, where *m* of *n* parties needs to sign a transaction to be valid.

Using Ocean Protocol, holding and transferring data access rights can ultimately be reduced to hold a private key of an Ethereum account.

4.5 Data exchanges

Data access rights in the form of fungible or non-fungible tokens can be exchanged and traded on supporting token exchanges. As of now, there are no centralized exchanges that support this, but on decentralized exchanges, the data token exchange is already operational because of the compatibility to ERC standards. Data tokens can be swapped for any ERC-based cryptocurrency on decentralized exchanges like Uniswap [27] or Balancer [28]. Said services are based on smart contracts that manage the escrow and release of tokens during an exchange. Because they are permissionless as well, it is possible for everyone to create any data token pair and to swap data tokens to any other data token or ERC-based currency. To participate, one needs only a wallet holding a public/private Ethereum key pair.

4.6 Data marketplaces

A data marketplace is a directory of data services. Providers can list their data assets and algorithms on the marketplace. Data consumers can search for data services and buy data access. The pricing of data assets can be fixed or dynamic. The functionality is governed by smart contracts, that allow data access once payment is completed. Data access can be of the following forms:

- download (of non-PII assets)
- stream (of non-PII assets)
- “compute-to-data”

The data provider chooses which access methods to allow once he registers a dataset on the marketplace. Depending on his choice, different contracts and tools are deployed in the background.

To sum it up, data tokens represent access rights to a data set. Data tokens can be held, transferred, exchanged and traded with any Ethereum-compatible software or device. Following that, with Ocean Protocol, it becomes possible to transfer, exchange and trade data access rights on a decentralized and permissionless network.

4.7 Data privacy

With Ocean Protocol data owners can keep control over their data sets by using a solution similar to federated machine learning which is in the Ocean vocabulary called “compute-to-data”. Given this concept, a dataset never leaves its location. Once data access is sold on a marketplace, the machine learning algorithm is brought to the data and executed on-premise. Hence, the data provider needs to deploy and run a software container next to his private data, where the algorithm is executed. The main difference in relation the federated learning is that there is no central authority that orchestrates the process. Smart contracts manage the compute-to-data functionality in a decentralized and trustless fashion. Compute-to-data will be analyzed in a dedicated chapter after introducing the full Ocean architecture.

4.8 License

Ocean Protocol is developed by BigchainDB GmbH [2] under the oversight of Ocean Protocol Foundation (OPF) [3]. The Ocean Protocol reference marketplace [29] was made publicly available on 27.10.2020 [30]. All tools are open-source and available under Apache 2.0 license on Github [4]. The license allows to freely use, modify and distribute the software as long as modifications and distributions contain a copy of the license, a list of modifications and a reference to the copyright owner.

As of December 2020, Ocean Protocol is still under development with most components being in beta status. A production-ready release is expected to be available in spring 2021. All Ocean Protocol tools used in this work were cloned or forked and modified from Ocean Protocols Github repository [4].

5. Deep Dive

After introducing the general concepts, this section presents the full Ocean Protocol toolkit and architecture.

5.1 Architecture

The Ocean Protocol architecture consists of three main layers: smart contracts, libraries/middleware and application layer. From a functionality perspective, it can be divided into data token & access control, market tools, metadata tools, and external tools. [31]

a) **Application layer**
The top layer holds applications like data marketplaces and data wallets, as well as related tools for the management of data tokens, data markets, data exchanges and metadata.

b) **Libraries / Middleware layer**
Middleware includes the Ocean Protocol JavaScript, Python and React libraries and metadata storage. They provide efficient utilities to the application layer so that applications do not directly need to interact with the blockchain and smart contract layer, which is comparatively slow and costly. The layer moreover accommodates Ocean Provider, which supports the consume of data assets and a container component executed by data providers to allow for on-premise computing. It is essential for allowing remote machine learning using "compute-to-data" functionality.

c) **Smart Contract layer**
Blockchain and smart contract functionality is located in the base layer. It contains a data token factory and data token templates that are invoked once a new dataset is registered. Furthermore, it enables static and dynamic pricing of datasets. In case a data provider opts for dynamic pricing, market liquidity is needed to allow quick purchases without causing drastic changes in the assets price. To provide market liquidity, a liquidity pool is instantiated from the liquidity pool factory and linked to the data token. In addition, because metadata is partly stored on-chain, it holds contract functionality to store metadata in smart contracts.
As Ocean Protocol is deployed on the Ethereum main net, it is based and reliant on a few lower layers (like the Ethereum blockchain with its P2P-network, consensus, EVM, and ledger state). They are not part of the Ocean Protocol toolkit and not shown here.

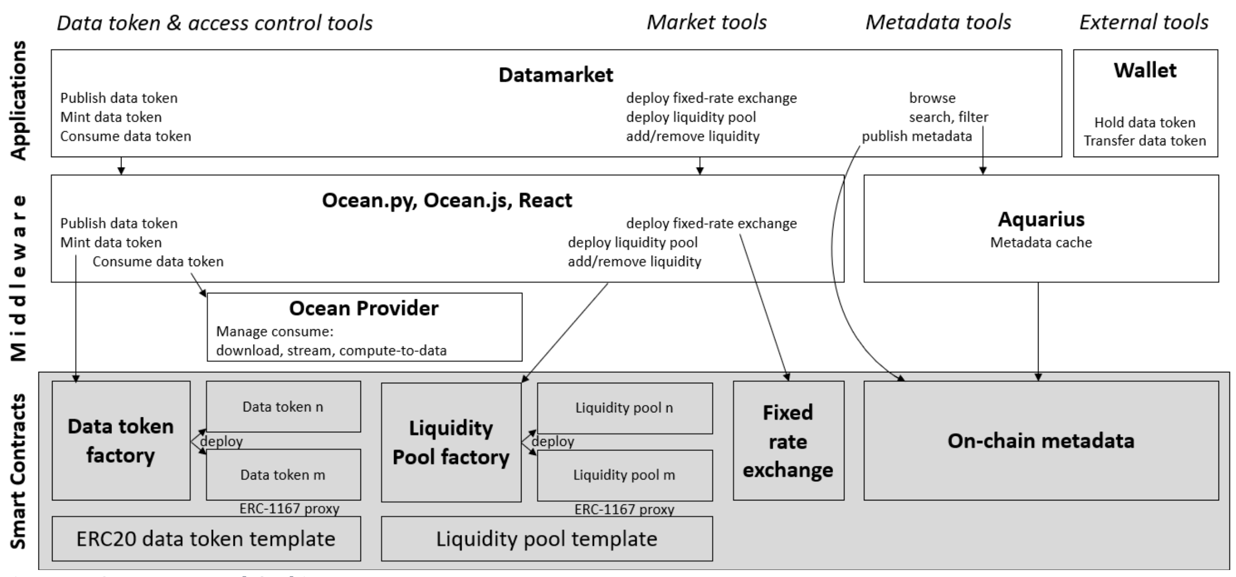


Figure 2: Ocean Protocol Architecture

5.2 Smart contracts

Ocean Protocol's core functionality is entirely based on smart contracts. To enable data tokens, token transfers, token exchange and payment, there exist the following contracts:

- Ocean token contract
- Metadata contract
- Data token factory contract
- Data token contracts
- Fixed-rate exchange contract
- Liquidity pool factory contract

Like explained before, a smart contract is a collection of code (functions) and data (state) that resides on the blockchain. The code is compiled into bytecode, and its content and state are written into a blockchain transaction. The program can be triggered by blockchain transactions and is then executed on every node of the network. In the following chapter, we introduce each contract and describe its most important functionality. States and functions irrelevant to the topic of privacy-preserving data exchange will be ignored.

5.2.1 Ocean Token Contract

The native Ocean token is an ERC-20 utility token used to buy, sell and curate data assets as well as to provide liquidity when using the reference implementation developed and provided by Ocean Protocol Foundation. Its use is not mandatory when deploying a marketplace and the underlying infrastructure. The native Ocean token or any ERC-20 token with equivalent properties can be used at the contract level. However, because the Ocean token is already deployed on all relevant Ethereum networks, it is convenient to use, and we will do so for testing. To avoid confusion, we refer to the token as a 'payment token' from now on.

Contract functions:

transfer(addressTo, value)	Transfers a specified number of tokens to a specified address.
approve(addressFrom, value)	Approves to spend the number of tokens from a specified address.
transferFrom(addressFrom, addressTo, value)	Transfers tokens from an address to another.
totalSupply()	Returns the total number of tokens in existence.
balanceOf(address)	Returns the balance of a specified address.
allowance(address, address)	Checks the number of tokens that an owner allowed to a spender.

Addresses:

Network	Address
Ethereum main-net	0x967da4048cD07aB37855c090aAF366e4ce1b9F48
Rinkeby test-net	0x8967BCF84170c91B0d24D4302C2376283b0B3a07
Ropsten test-net	0x5e8DCB2AfA23844bcc311B00Ad1A0C30025aADE9

5.2.2 Metadata Contract

The metadata contract is used to (partly) store the metadata of every data asset published on Ocean Protocol. The exact implementation of metadata storage is described in Section 5.3.

Contract state:

Variable	Type	Description
dataToken	address	The address of the data token belonging to the data asset.
created/updated by	address	The address of the market participant who created or updated a data asset.
flags	bytes	Indicates special flags associated with metadata.
data	bytes	A unique reference to the metadata (DID).

Contract functions:

create(dataToken, flags, data)	Creates/publishes a new metadata entry on chain.
update(dataToken, sender, flags, data)	Allows the data token creator to update a metadata entry.

Addresses:

Network	Address
Ethereum main-net	0x1a4b70d8c9DcA47cD6D0Fb3c52BB8634CA1C0Fdf
Rinkeby test-net	0xFD8a7b6297153397B7eb4356C47dbd381d58bFF4
Ropsten test-net	0x3cd7Ef1F207E1a46AAd7D5d7F5f0A5cF081Fc726

5.2.3 Data Token Factory Contract

The data token factory contract creates a data token and its corresponding contract when called. It is usually triggered when a data provider registers a new data asset on the marketplace to create a related data token. Technically, it instantiates an ERC-20 token contract from an ERC-20 data token template through an ERC-1167 proxy contract.

ERC-1167 is a standardized solution to deploy contract clones in a simple, cheap and immutable way [32]. Because each dataset has its own token and consequently its own token contract, a lot of very similar contracts are to be expected once Ocean Protocol is used at scale. Without using a proxy contract, each of these contracts would need to be compiled and deployed separately, which is costly and creates on-chain redundancy because the same bytecode would need to be stored for each copy. On-chain storage is costly. To give an example, storage of one megabyte (1MB) of data on the Ethereum blockchain costs about 6,000 EUR as of November 2020[4]. The proxy contract relays every incoming transaction to a reference contract and stores minimal data to circumvent this. Every proxy is a replica of the reference contract but serves a different token.

Contract state:

Variable	Type	Description
tokenTemplate	address	The address of the ERC-20 data token template.
communityFeeCollector	address	The address of the community fee collector. Marketplaces can take fees (usually 0.3%) from marketplace participants to fund their operation. This is not relevant in our context and will be ignored.
currentTokenCount	uint256	The number of data tokens minted.

Contract functions:

constructor(tokenTemplate, communityFeeCollector)	The constructor is called at contract deployment given the ERC-20 token template and an address to collect community fees.
createToken(blob, name, symbol, cap)	Creates a new data token proxy contract given its name, symbol and quantity.

Addresses:

Network	Address
Ethereum main-net	0x57317f97E9EA49eBd19f7c9bB7c180b8cDcbDeB9
Rinkeby test-net	0x3fd7A00106038Fb5c802c6d63fa7147Fe429E83a
Ropsten test-net	0x6ebcCa6df2CAba986FCF44E64Ee82251c1455Dcc

[4] To store a 256 bit word on the Ethereum blockchain costs 20,000 gas fees [33]. A megabyte is thus 655,360,000 gas. The average gas price as of late 2020 is about 20 gwei (0.00000002 ETH), so a megabyte of storage costs about 13 ETH which is circa 6,000 EUR (at 470 EUR per ETH in December 2020). It must be noted that prices are volatile. Gas prices change continually depending on blockchain utilization. ETH prices are subject to market volatility.

5.2.4 Data Token Contract

The data token contract is created by the data token factory. Like explained before, it is instantiated from an ERC-20 data token template through an ERC-1167 proxy contract. A data token contract is unique to each data asset registered on Ocean Protocol.

Contract state:

Variable	**Type**	**Description**
name	string	Data token name.
symbol	string	Data token symbol.
blob	string	A reference to the metadata store.
cap	Uint256	Number of tokens.
decimals	Uint8	Number of supported decimal points.
minter	address	Specifies the address that is allowed to mint tokens.

Contract functions:

transfer(addressTo, value)	Transfers a specified number of tokens to a specified address.
approve(addressFrom, value)	Approves to spend the number of tokens from a specified address.
transferFrom(addressFrom, addressTo, value)	Transfers tokens from an address to another.
mint(address, value)	Allows to mint data tokens. Can only be called by the address specified in the minter variable, e.g. the contract owner.
startOrder(address, amount, serviced, marketFeeCollector)	The function is called by the data consumer to initiate data asset consumption. It transfers one data token to the seller.
finishOrder(orderTxId, address, amount, serviced)	The function is called by Ocean Provider only if there is a partial or full refund (in case the data to be sold is missing or corrupted).

5.2.4 Fixed Rate Exchange Contract

The fixed-rate exchange contract allows data consumers to exchange payment tokens for data tokens using a fixed exchange rate. Swapping the token is equivalent to a token transfer in both directions with an escrow functionality in the middle, managing the exchange. A fixed-rate exchange contract is always deployed when a data asset is published, and the publisher opts for fixed pricing.

Contract state:

Variable	Type	Description
active	boolean	Indicates whether the exchange is active.
exchangeOwner	address	Specifies the exchange owner.
dataToken	address	Refers to a data token contract address.
paymentToken	address	Refers to a payment token contract address.
fixedRate	uint256	The exchange rate between data and payment tokens.

Contract functions:

create(paymentToken, dataToken, fixedRate)	Creates a new exchange pair between a payment token and data token and sets a fixed exchange rate.
swap(exchangeId, dataTokenAmount)	Does an atomic swap with the number of data tokens to be exchanged.

Addresses:

Network	Address
Ethereum main-net	0x608d05214E42722B94a54cF6114d4840FCfF84e1
Rinkeby test-net	0xeD1DfC5F3a589CfC4E8B91C1fbfC18FC6699Fbde
Ropsten test-net	0xA7a711A09396DF82D9be46A26B48BafdB9BB4fA6

5.2.5 Liquidity Pool Factory Contract

The liquidity pool factory contract instantiates a new liquidity pool that allows data consumers to exchange payment tokens for data tokens using a dynamic exchange rate. A liquidity pool contract is always deployed when a data asset is published, and the publisher opts for dynamic pricing. The pool holds data tokens and payment tokens. The details and price dynamics are described in Section 5.4. Technically, the smart contract is again deployed by cloning from a reference contract using an ERC-1167 proxy to minimize on-chain redundancy and network fees.

Contract state:

Variable	**Type**	**Description**
dataTokenAddress	address	Refers to a data token contract address.
dataTokenAmount	uint256	The number of data tokens in the pool.
dataTokenWeight	uint256	The ratio of data tokens in relation to payment tokens.
paymentTokenAddress	address	Refers to a payment token contract address.
paymentTokenAmount	uint256	The number of payment tokens in the pool.
paymentTokenWeight	uint256	The ratio of payment tokens in relation to data tokens.

Contract functions:

constructor(address)	The constructor is called at contract deployment given the liquidity pool template address.
newPool()	Deploys a new liquidity pool proxy contract.
setup(dataTokenAddress, dataTokenAmount, dataTokenWeight, baseTokenAddress, baseTokenAmount, baseTokenWeight, swapFee)	Setups a liquidity pool with data and payment tokens and their respective weights.
getSpotPrice(tokenAddressIn, tokenAddressOut)	Calculates and returns the spot price based on the number of data and payment tokens.
swap(callerAddress, tokenAddressIn, tokenAddessOut, tokenAmountIn, tokenAmountOut)	Swaps a specified number of data/payment tokens from and to a callers address.
joinSwap(callerAddress, tokenAddressIn, tokenAmountIn)	Allows a caller to add liquidity into the pool.
exitSwap(callerAddress, tokenAddressOut, tokenAmountOut)	Allows a caller to remove liquidity from the pool.

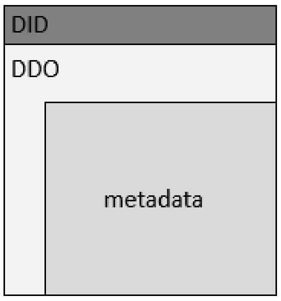
Addresses:

Network	**Address**
Ethereum main-net	0xbe0083053744ECb871510C88dC0f6b77Da162706
Rinkeby test-net	0x53eDF9289B0898e1652Ce009AACf8D25fA9A42F8
Ropsten test-net	0x75be6e18c80A487C8b49663bf14f80A6495045B2

5.3 Metadata

Metadata is essential to be able to search, find and discover data assets effectively. An assets metadata is set by its publisher. Metadata consists of the following attributes, all of which are objects:

Attribute	Required	Description
(1) main	yes	Main attributes used to calculate the service checksum.
(2) curation	(remote)	Curation attributes.
(3) additionalInformation	no	Optional attributes.
(4) encryptedFiles	(remote)	Encrypted string of the attributes.main.files object.
(5) encryptedServices	(remote)	Encrypted string of the attributes.main.services object.



Together the attributes (1 - 5) form a Decentralized Descriptor Object (DDO) that is a JSON document holding all the metadata of a data asset [34]. Each DDO is referenced by a decentralized identifier (DID) [35], which is a hex string that is calculated by hashing an assets DDO checksum using SHA3-256. [36] Hence, a DID is unique and references its corresponding DDO.

Note: Descriptive labels are not part of the reference design, but we use them here to facilitate understanding.

- Metadata object:	1 - 5
- Main attributes:	m1 - m7
- File attributes:	f1 - f13
- Additional attributes:	a1 - a10

The main attribute object (1) holds a list of the following attributes (m1-m7). Because main attributes are used to calculate the service checksum, they cannot be modified after creation.

Main attributes (1) [34]

Attribute	Required	Description
(m1) name	yes	Descriptive name or title of the asset.
(m2) type	yes	Type of asset. ("dataset", "algorithm").
(m3) dateCreated	yes	The date on which the asset was created by the originator, ISO 8601 format, UTC.
(m4) datePublished	(remote)	The date on which the asset DDO is registered into the metadata store (Aquarius). Is set automatically when publishing.
(m5) author	yes	Name of the entity generating the data asset.
(m6) license	yes	Short name referencing the license of the asset (e.g. Public Domain, CC-0, CC-BY, No License Specified, …).
(m7) files	yes	Array of file objects. Contains all file metadata, as shown in the next table.

For each file, a **file attribute object** (m7) is stored. [34]

Attribute	**Required**	**Description**
(f1) url	(local)	Content URL. Supports http(s):// and ipfs:// URLs. The url is only held locally and omitted when publishing.
(f2) name	no	File name.
(f3) Index	yes	Index number starting from 0 of the file.
(f4) contentType	yes	File format.
(f5) Checksum	no	Checksum of the file (i.e. MD5,SHA). Format specified in checksumType. If it is not provided it cannot be validated if the file was not modified after registering.
(f6) checksumType	no	Format of the provided checksum. Can vary according to storage service (i.e AWS, Azure).
(f7) contentLength	no	Size of the file in bytes.
(f8) Encoding	no	File encoding (e.g. UTF-8).
(f9) Compression	no	File compression (e.g. no, gzip, bzip2, …).
(f10) Encrypted	no	Boolean. Is the file encrypted? If is not set, it is assumed the file is not encrypted.
(f11) encryptionMode	no	Encryption mode used. Just valid if encrypted=true.
(f12) resourceId	no	Remote identifier of the file in the external provider. It is typically the remote id in the cloud provider.
(f13) attributes	no	Key-Value hash map with additional attributes describing the asset file. It could include details like the Amazon S3 bucket, region, ...

Selected additional attributes (3) [34]

Attribute	**Required**	**Description**
(a1) categories	no	A list of categories describing the asset.
(a2) tags	no	A list of keywords/tags describing the asset.
(a3) description	no	Additional description.
(a4) copyrightHolder	no	Copyright or IP holder.
(a5) links	no	Links, http(s):// or ipfs://
(a6) isLanguage	no	Language of the asset.
(a7) sla	no	Service level agreements.
(a8) updateFrequency	no	An indication of update latency - i.e. How often are updates expected (seldom, annually, quarterly, …), or is the resource static that is never expected to get updated.
(a9) termsOfService	no	Terms of Service.
(a10) structuredMarkup	no	A link to machine-readable structured markup (such as ttl/json-ld/rdf) describing the dataset.

Not all attributes are set by the publisher. Some are set automatically during the publishing process, like for example ‘datePublished’ (m4) which is indicated by the ‘remote’ tag. Others are only used locally, like ‘URL’ (f1).

Metadata storage
Because blockchain storage is expensive, Ocean stores minimal data on-chain. Only DID, DDO and the data assets URL (encrypted) are stored on-chain. This is done by issuing a transaction to the metadata smart contract that contains the following fields:

create {DID,{dataTokenAddress, encrypted[URL], DDO}, optional}

The 'URL' attribute (f1) which holds the original plain text URL is only held locally and not written into the on-chain DDO.

To separate the market application from the comparatively slow blockchain layer and to allow for additional metadata fields that exceed the size to be reasonably stored on-chain, Ocean Protocol uses additional off-chain metadata storage. The off-chain component is called 'Aquarius'. It retrieves metadata entries from the chain, caches them locally and manages additional information that complements the DDO. Storing the metadata on-chain costs about 1.5 - 3.5 EUR, depending on its size and current gas prices. It is paid once by the publisher. Afterwards, the metadata can be retrieved from the chain by anyone.

Metadata integrity
The concept of separating the identifier and the metadata has the advantage to be able to ensure its integrity. Because the DID is effectively a hash of the DDOs main attributes, modification of main metadata attributes is prevented by design. While a modification of non-main attributes is possible by calling the update() function of the metadata contract, an update of main attributes is only possible by deriving a new data asset using the create() function of the metadata contract with a new DID. Furthermore, the asset owner can choose to sign a specific integrity checksum corresponding to an asset. This allows a third party to validate that changes in non-main attributes were made or approved by the owner.

5.4 Pricing

Pricing of data assets can be fixed or dynamic. With fixed pricing, a smart contract with exchange functionality is deployed for every data asset that swaps a fixed number of ERC-20 payment tokens for one data token.

With dynamic pricing, a decentralized automated market is deployed for every dataset. The automated market is implemented with the help of a liquidity pool that holds ERC-20 compatible payment tokens and data tokens. The benefit of such a pool is that it can be implemented on a smart contract to offer automated market maker (AMM) functionality for the pricing of datasets. Hence, the pricing of datasets becomes possible without maintaining a traditional order book or other centralized components. It operates autonomously in a trustless environment.

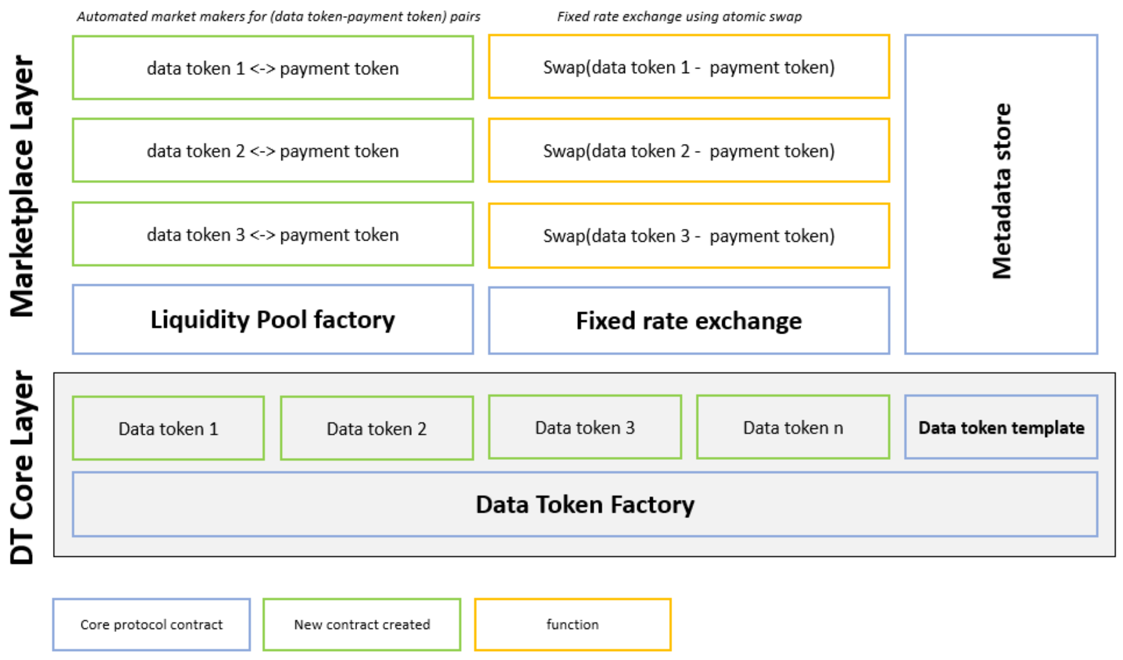

Automated market makers for (data token-payment token) pairs
Fixed rate exchange using atomic swap
Marketplace Layer
data token 1 <-> payment token
data token 2 <-> payment token
data token 3 <-> payment token
Liquidity Pool factory
Swap(data token 1 - payment token)
Swap(data token 2 - payment token)
Swap(data token 3 - payment token)
Fixed rate exchange
Metadata store
DT Core Layer
Data token 1
Data token 2
Data token 3
Data token n
Data token template
Data Token Factory
Core protocol contract
New contract created
function

Figure 3: Ocean Protocol data token exchange overview

The dynamic pricing of data is complex, and Ocean Protocol offers an in-depth solution using AMMs combined with liquidity staking. The implementation is based on Balancer [37], a decentralized and trustless protocol for programmable liquidity. This enables individuals and institutions to explore the value of their data and to let the market price it based on supply and demand. However, as this is not necessarily relevant to the scope of this thesis and to archieve the goal of privacy-preserving data marketplaces, we only give a brief introduction.

In addition, dynamic pricing serves another purpose: It enables the possibility of data curation. Data curation in Ocean Protocol is equal to liquidity staking. Liquidity providers are incentivized to search for data assets and to evaluate their quality and valuation as they can earn from transaction fees. Each time a data asset is sold, liquidity providers share a cut of the fees (usually 0.1%). In order to maximize their ROI, liquidity providers will likely direct their liquidity towards high quality, but undervalued data assets. To stake liquidity, a liquidity provider inserts tokens into a liquidity pool. By locking his tokens, a liquidity provider vouches for the quality of the data. He gains a share in the liquidity pool if other liquidity providers come to the same conclusion that the quality is equally good or better, or if they assess that the asset is still undervalued. On the other hand, he loses a share if the data asset turned out to be less in demand. The mechanism ensures that with enough data curators and over time, a fair market price is found. Data assets of a higher quality rise in visibility and valuation over time, while lower quality to worthless assets slowly disappear.

5.5 Access Control

Access to data assets can be of the types of

a. **Download**
The data consumer can download the data asset.

b. **Stream**
The data consumer can stream the data asset.

c. **Compute-to-data**
The data consumer purchases compute-only access that allows him/her to run remote compute jobs on the data, while the data stays on-premise. Compute-to-data functionality is crucial to enable privacy-preserving marketplaces because it ensures GDPR compliance. Hence it will be discussed in a dedicated chapter (5.6).

Access to data assets is managed by smart contracts. Given the properties of data tokens, it becomes possible to restrict access to a data asset (of any access type) in a decentralized and trustless manner based on various conditions:

- grant equal access for all token holders, but limit the number of possible accesses by minting a fixed amount of ERC-20 tokens
- ensure individualized or personalized access control by using ERC-721 tokens
- implement time-bound access
- restrict to one-time access
- allow perpetual access
- restrict the access methods (download, stream, compute access)
- allow read or write access
- group access

Once a data asset is bought by a consumer, the smart contract logic assesses who is allowed to access which resource. In case of the access types ‘download’ or ‘stream’, the contract grants access as soon as one data token arrives and the consumer called the startOrder() function of the data token contract. Once this is completed successfully, the Ocean Provider service loads the encrypted URL, decrypts it and relays it to the consumer.

On the other hand, when the data publisher restricted the access type to ‘compute’ a so-called ‘Service Execution Agreement’ (SEA) is deployed. A SEA is an agreement between a data publisher, consumer and a verifier that is implemented on a smart contract. It specifies the conditions to be fulfilled to access a data asset, what data assets need to be delivered and the rewards for meeting the conditions. A SEA is created upon every purchase of compute access. It holds the DID, consumer and provider addresses, payment information and is signed by the consumer and validated by the provider. An algorithm can be included in the form of a payload. Alternatively, a pre-published algorithm can be used by referencing its DID. Moreover, it includes an escrow function which manages the payment. A consumer locks tokens in escrow. If the computation terminates successfully, the compute provider is allowed to withdraw those tokens, otherwise (timeout, unsuccessful) they are returned to the consumer. Once the compute job terminates, the results can be accessed via a URL, or they are published in the form of a new data asset, referenced by a new DID.

5.6 Compute-to-data

Compute-to-data allows for privacy-preserving data sharing and is a core feature and foremost advantage of Ocean Protocol. The idea behind compute-to-data is to keep the data on-premise and to allow data consumers to run remote compute jobs on the data. Data owners keep full control as the data never leaves their location. [38] Compute-to-data is directly integrated into data marketplaces, where data providers can opt to restrict the access type to "compute only". Once data access is sold on a marketplace, the compute job is brought to the data and executed on-premise. This is especially suitable to train machine learning models remotely, and the concept is very similar to federated ML. Using Ocean Protocol, data consumers can train their machine learning models across many data sets of different data providers while ensuring compliance with privacy regulations. Furthermore, because compute-to-data on Ocean Protocol operates in a trustless and decentralized manner, the central authority that was needed to orchestrate the process of federated ML is no longer necessary.

Compute-to-data adds a new role to the data marketplace, called "compute provider". Compute providers sell computation on data instead of the data itself. The compute provider can be the same entity as the data provider. From a security perspective, it is recommended for data providers to take over both roles to keep full control over the data. The compute provider needs to deploy and run a software container next to the data where the computation is executed.

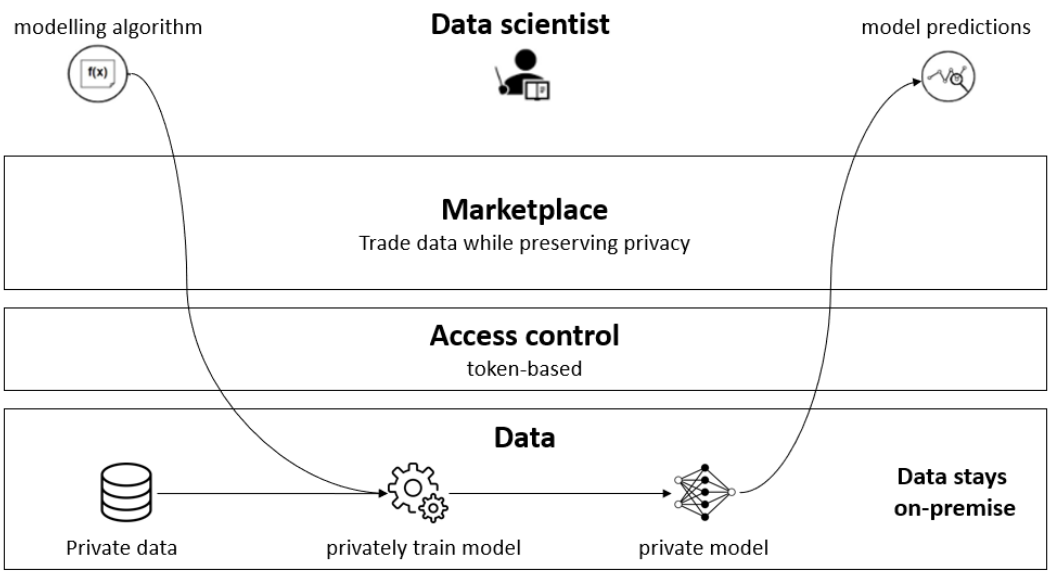


Figure 4: Compute-to-data

Architecture

Compute-to-data introduces two new components to the collection Ocean Protocol tools:

- **Operator Service**
  The Operator service manages the compute-to-data workflow, communicates with the data provider, and performs the computation.
- **Operator Engine**
  The Operator Engine is a backend service that orchestrates and manages the compute infrastructure.

Both use a Kubernetes K8s cluster[5] for the compute execution and are maintained by the compute provider. The compute provider needs to deploy the containerized applications in his/her infrastructure, provide resources (CPU, memory, storage, bandwidth) and set up read-only access to the storage holding the data where access is sold to. The compute provider decides what exact compute resources should be available to compute consumers, supporting JavaScript and Python runtimes as of early 2021.

Data assets of the compute type have additional metadata attributes. They are necessary to orchestrate and control the remote computation and describe the language, runtime, version, container and related information. [34]

Attribute	**Required**	**Description**
(1) language	no	Language used to implement the algorithm
(2) format	no	Packaging format of the algorithm
(3) version	no	Version
(4) container	yes	Object describing the docker container image

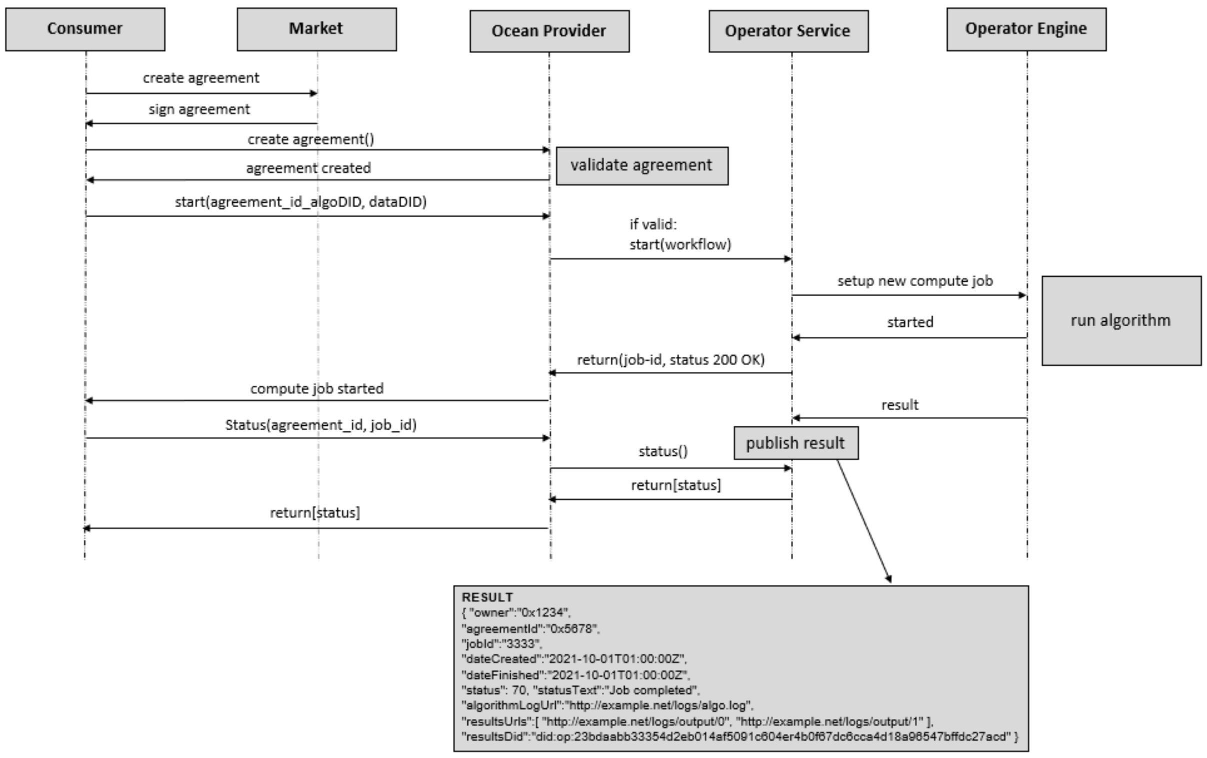
The container object (4) has the following attributes:

Attribute	**Required**	**Description**
entrypoint	yes	The command to execute or script to run inside the docker container image
image	yes	Name of the docker container image
Tag	yes	Tag of the docker container image

Starting a compute job

Once a compute service is bought by a consumer, a Service Execution Agreement (SEA) [40] is created on-chain. It holds the DID, consumer and provider addresses, payment information and is signed by the consumer and validated by the provider. An algorithm can be included in the form of a payload. Alternatively, a pre-published algorithm can be used by referencing its DID. Moreover, it includes an escrow function which manages the payment. A consumer locks tokens in escrow. If the computation terminates successfully, the compute provider is allowed to withdraw those tokens, otherwise (timeout, unsuccessful) they are returned to the consumer. Once the compute job terminates, the results can be accessed via a URL, or they are published in the form of a new data asset, referenced by a new DID.

[5] "A Kubernetes cluster is a set of nodes that run containerized applications. Containerizing applications packages an app with its dependences and some necessary services. They are more lightweight and flexible than virtual machines. Kubernetes clusters allow containers to run across multiple machines and environments and are not restricted to a specific operating system. They are able to share operating systems and run anywhere." [39].



```
RESULT
{ "owner":"0x1234",
"agreementId":"0x5678",
"jobId":"3333",
"dateCreated":"2021-10-01T01:00:00Z",
"dateFinished":"2021-10-01T01:00:00Z",
"status": 70, "statusText":"Job completed",
"algorithmLogUrl":"http://example.net/logs/algo.log",
"resultsUrls":[ "http://example.net/logs/output/0", "http://example.net/logs/output/1" ],
"resultsDid":"did:op:23bdaabb33354d2eb014af5091c604er4b0f67dc6cca4d18a96547bffdc27acd" }
```

Figure 5: Starting compute-to-data remote compute jobs [41]

After completion of the compute job, the results can be retrieved via URL or DID if the consumer requested to publish the results as a separate asset.

Trusting algorithms

With compute-to-data, it is essential that algorithms do not act maliciously. Algorithms leaking sensitive information, PII or returning the data itself defeat the purpose of compute-to-data and pose a critical risk. It is the data owner's responsibility to choose which algorithms to trust. Because of the trustless environment, anyone can buy compute access. Hence, the selection and approval of algorithms need to be governed and restricted effectively. A narrow implementation of vetted algorithms on the Ocean Protocol reference marketplace (GUI) is currently in development by the Ocean Protocol Foundation and a solution towards extensive GUI-based control is planned. As of January 2021, it is possible to restrict the supported algorithms to pre-published known (and vetted) ones by registering them as a data asset and referencing the DID inside the compute SEA. This is feasible for common use cases and computations.

As the remote computation and selection of algorithms pose potential risks, the topic is further discussed in Section 7.1 (Evaluation/Security).

5.7 Protocol flow

5.7.1 Publish a dataset

Like described before, a data marketplace is a directory of data services where providers can list their data sets and algorithms on the marketplace. Data consumers can search for data services and buy data access. To publish a dataset on the marketplace, the data seller needs to provide the following main metadata attributes. Optionally the publisher can include any other metadata attribute described in Section 5.3.

- provide a title
- provide a description
- provide a URL to specify where the data is stored
- specify the access type (download, stream, compute)
- provide the name of the author
- provide the license

Alice publishes a data asset:

1. **Preparation**
   - Alice stores the data asset "asset1" on her server, to be accessed at 'URL: alice.com/data/asset1.csv'
   - Alice has an Ethereum address (for example 0x112935) and ETH to pay for gas fees
   - Alice visits any data marketplace based on Ocean Protocol. She can rely on their Ocean Provider service (for example *ocean-provider.com)* or run her own
   - Alice provides a title, description, URL, access type, author and license
   - Alice clicks "publish"

2. **Data token contract creation**

   Once the publication is initiated, the marketplace application triggers the creation of a dedicated data token contract to the Ethereum network. The information is forwarded by the respective libraries to the 'data token factory contract', which instantiates an ERC-20 data token contract from the ERC-20 data token template. This is done through an ERC-1167 proxy contract.

   The resulting token holds the following information

name	string	Data token name
symbol	string	Data token symbol
blob	string	A reference to the metadata store
cap	Uint256	Number of tokens
decimals	Uint8	Number of supported decimal points
minter	address	Specifies the address that is allowed to mint tokens

   It is generated by issuing a transaction to the data token factory contract. As an example, the following transaction mints 100 data tokens with symbol 'ADT-1' and name 'alice-datatoken', that reference to 'example-metadata-store.com' for metadata storage.

   dataTokenFactory.createToken(blob:example-metadata-service.com,
   name: alice-datatoken, symbol: ADT-1, cap:100)

   The token contract address is assigned by Ethereum. We assume *0x1234* for this example.

On-chain registration

Metadata is stored on-chain. To initiate the process, the marketplace calls the create() function on behalf of Alice. It creates a DDO document holding the associated metadata [34] and calculates its DID. The URL provided by Alice that specifies where the data asset can be retrieved (url:alice.com/data/asset1.csv) is separated from the metadata and encrypted. Moreover, the Ocean Provider URL and data token Ethereum address is added.

Ocean_assets.create(metadata, ocean-provider.com/api/consume, 0x1234)

The URL decryption key is stored by Ocean Provider, which is usually operated by the same entity that runs the data marketplace (in case of a simple download) or the entity that runs the compute-to-data container. If desired, Alice could run her own Ocean Provider.

DID, DDO and the encrypted URL are stored on-chain by issuing a transaction to the metadata smart contract:

*DDO.create({DID,{dataTokenAddress: 0x1234, encrypted[*alice.com/data/asset1.csv*], DDO}, optional}*)

3. **Setting a price**
   Afterwards, Alice needs to specify whether the dataset should be priced in a fixed or dynamic way.

   a. **With fixed pricing**, Alice chooses a price (denominated in any ERC-20 compliant token and denominated in Oceans native token when using the reference implementation). Then she specifies the number of tokens to be minted and invokes the mint function of the data token contract. Afterwards, she issues a transaction to the fixed-rate exchange contract to create an exchange for her data-token/payment-token pair. Alice then approves to send the minted data tokens to the exchange contract.

      As an example, the following transactions mint 900 data tokens with symbol ‘ADT-1’ and name ‘alice-datatoken’ (referenced by the token address 0x1234) and create a 1:1 exchange contract between data tokens (0x1234) and payment tokens (0x4444). Then Alice approves to insert 900 data tokens into the exchange contract.

      dataToken.mint(address: 0x1234, value:900)
      fixedRateExchange.create(paymentToken: 0x4444, dataToken: 0x1234, fixedRate:1-1)
      dataToken.approve(addressFrom: 0x112935, value:900)

b. **With dynamic pricing**, a decentralized automated market based on a smart contract is created for every dataset. To do this, Alice specifies the number of tokens to be minted and invokes the mint function of the data token smart contract. Following that, a liquidity pool is created by calling the liquidity pool factory contract. The pool holds the minted data tokens and any ERC-20 compliant token or Oceans native token when using the reference implementation. Technically, the automated market smart contract is again deployed by cloning from a reference contract using an ERC-1167 proxy.

   As an example, the following transactions mint 900 data tokens with symbol ‘ADT-1’ and name ‘alice-datatoken’ (referenced by the token address 0x1234) and create an equally weighted (1-1) liquidity pool contract between data tokens (0x1234) and payment tokens (0x4444). Then Alice approves to insert 900 data tokens into the liquidity pool.

```
dataToken.mint(address: 0x1234, value:900)
liquidityPoolFactory.newPool()
liquidityPool.setup(dataToken: 0x1234, dataTokenAmount: 900, dataTokenWeight:
0.5, paymentToken: 0x4444, paymentTokenAmount:900, paymentTokenWeight:0.5,
swapFee:1)
dataToken.approve(addressFrom: 0x112935, value:900)
```

After following the four steps, the data asset is registered on-chain and ready to be consumed. The data asset can be discovered and bought by anyone.

The Ethereum gas fees to publish a dataset (including metadata and all necessary contracts) with fixed pricing are about 15 EUR and with dynamic pricing about 20 EUR as of December 2020. The fee is paid once per data asset registered on-chain by the publisher.

5.7.2 Consume a dataset

As soon as a dataset is published, it can be bought and accessed by data consumers. To consume a dataset, the consumer needs to follow the steps:

Bob consumes a data asset:

1. **Find a dataset**
   As Ocean Protocol is open-source and permissionless, anyone can deploy data marketplaces on top. The Ocean Protocol Foundation runs a reference marketplace called "Ocean Market" [29], which can be used to explore its capabilities and for testing. To find a dataset, the potential buyer discovers suitable marketplaces and searches for datasets. To simplify the selection, data providers can include a sample file which shows the data structure.

2. **Buy the data token**
   Once Bob finds a suitable data asset, he buys the corresponding data token. Data tokens can be traded on any ERC-20 compatible exchange or bought directly on the data marketplace. This is done by connecting an ERC-20 wallet to the marketplace, approving the transaction, and swapping the required amount of payment tokens for at least one data token. Swapping the token is equivalent to a token transfer in both directions with an escrow contract in the middle, managing the exchange. Once swapped, Bob can hold the data token in his wallet for later use or immediately spend it to access the dataset.

   - for fixed-price assets:
     fixedRateExchange.swap(exchangeId: 9999, dataTokenAmount:1)

   - for dynamically prices assets:
     liquidityPool.swap(BobAddress, paymentToken: 0x4444, dataToken: 0x1234, amountIn:1, amountOut:1)

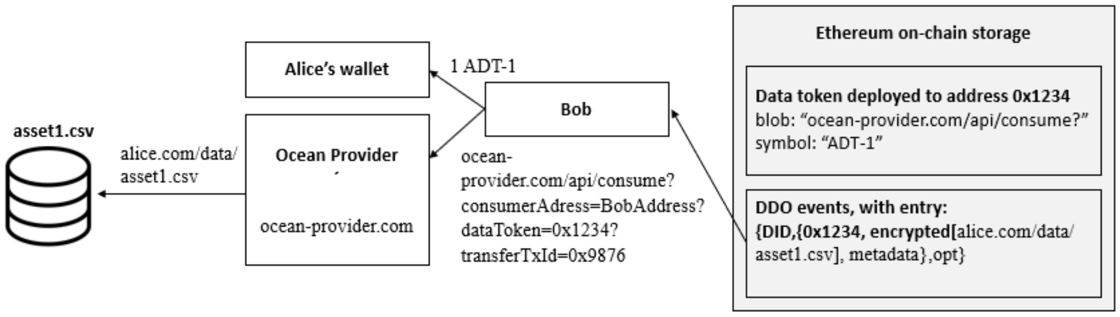
3. **Access**
   To be granted access to a dataset, Bob needs to hold at least one data token and to transfer the token to the data publisher (Alice). Bob unlocks access by calling the data tokens startOrder() function. He needs to provide the respective arguments of the data asset he wants to access, which he finds in the DDO in the assets metadata contract. The function transfers one data token to Alice's wallet.

   startOrder(dataToken: 0x1234, amount: 1.0, serviced: …, marketFeeCollector: …)

   After the transactions are confirmed, Bob needs to note the transaction id (for example 0x9876) to be able to prove his payment to Alice to Ocean Provider. Bob calls the Ocean Provider service. He finds the address (in this case 'https://ocean-provider.com/api/consume') in the data assets metadata entry.

   https://ocean-provider.com/api/consume?consumerAdress=BobAddress?dataToken=0x1234?transferTxId=0x9876

   In case of buying access to a compute-to-data service, tokens are not directly sent to Alice. When buying compute access, Bob sends his data token to an escrow contract and it is held there until the computation terminates.



Ethereum on-chain storage
Alice's wallet
1 ADT-1
Bob
Data token deployed to address 0x1234
blob: "ocean-provider.com/api/consume?"
symbol: "ADT-1"
asset1.csv
alice.com/data/asset1.csv
Ocean Provider
ocean-provider.com
ocean-provider.com/api/consume?consumerAdress=BobAddress?dataToken=0x1234?transferTxId=0x9876
DDO events, with entry:
{DID,{0x1234, encrypted[alice.com/data/asset1.csv], metadata},opt}

Figure 6: Consume a data asset

Ocean Provider checks if the transaction is confirmed and indeed one data token has been transferred to Alice. If yes, access is granted.

a. **Download**
The Ocean Provider component fetches the encrypted data asset URL from the chain, decrypts it using the corresponding key it holds, accesses the URL and relays the content to the data consumer (Bob). From the perspective of the consumer, the dataset is delivered directly by the marketplace. The URL is not disclosed.

b. **Stream**
Access to streams is similar. The Ocean Provider component fetches the encrypted streaming URL from the chain, decrypts it using the corresponding key it holds and creates a temporary URL to relay the streaming content to the data consumer. From the perspective of the consumer, the stream is delivered directly by the marketplace. The original URL is not disclosed.

c. **Compute-to-data**
A Service Execution Agreement (SEA) is created on-chain which holds the DID, consumer and provider addresses, payment information and is signed by the consumer and validated by the provider. An algorithm can be included in the form of a payload, or a pre-published algorithm is referenced by pointing to its DID. The algorithm is then brought to the compute container that has read access to the private data asset and is executed inside the capsuled environment. If the computation terminates successfully, the compute provider is allowed to claim the tokens escrowed in step 3 from the SEA, otherwise (timeout or unsuccessful) they are returned to the consumer. Once the compute job terminates, the results can be accessed via a URL, or they are published in the form of a new data asset, referenced by a new DID.

6. Deployment of a privacy-preserving data marketplace prototype

To verify and test the concept of a privacy-preserving data marketplace based on Ocean Protocol in practice, we decided to deploy a prototype. As the compute-to-data functionality is a prerequisite to guarantee compliance with privacy law, the prototype should include all Ocean Protocol components required to test compute-to-data under realistic conditions. The prototype includes the connection to a permissionless blockchain, the possibility to publish and consume data assets, the handling of payments, as well as the execution of remote compute jobs on private data sets.

6.1 Prototype architecture

From a high-level perspective, the prototype consists of three parts implemented as virtual machines (VM) that represent the market participants and the underlying infrastructure. The data provider and data consumer are on both ends, and the software tools necessary to enable the market functionality in the middle. The tools consist of the connectivity to the blockchain layer, metadata management, storage and a marketplace. To model real-world conditions and to test the permissionless approach, each part is separated and only connected to the internet.

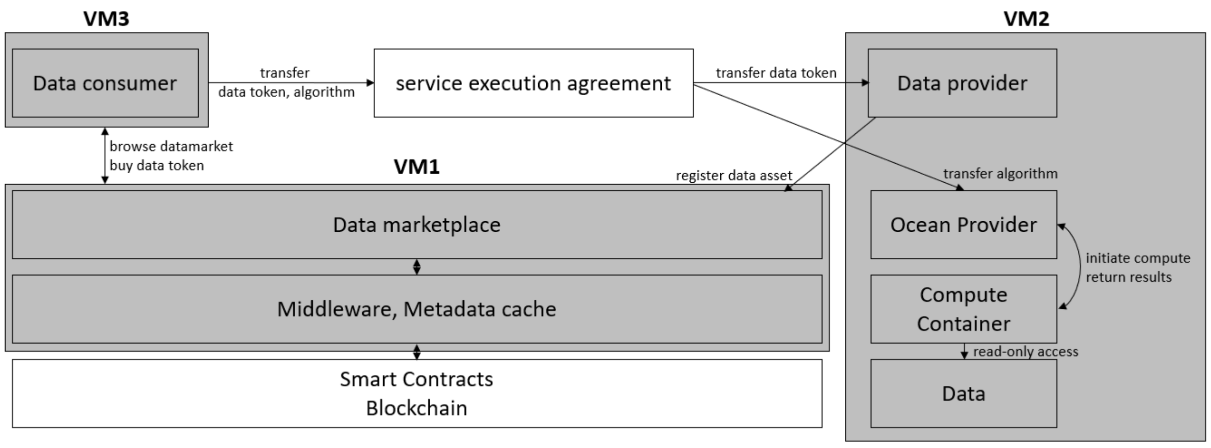


Figure 7: Prototype architecture

6.1.1 Infrastructure

The infrastructure VM holds all tools necessary to host a data market: This includes tools to

- Connect to the Ethereum networks (main and test nets)
- Deploy smart contracts to store metadata on-chain, create data tokens, enable static or dynamic pricing of data assets, and enable payment and escrow functionality
- Query and write on-chain metadata
- Cache metadata
- Publish datasets
- Consume datasets
- Manage payment between participants

Blockchain and smart contracts

The permissionless approach allows it to leverage existing smart contracts by the Ocean Protocol Foundation or to deploy our own contracts. Using existing contracts, one can directly publish data assets, which is a trade-off between convenience and control. Contracts are available on Ethereum main-net, Rinkeby test-net and Ropsten test-net [42].

Metadata

Reading and caching of metadata to and from the chain is managed by a component called ‘Aquarius’ [43]. It is provided with the contract address of the metadata smart contract and regularly caches all metadata stored on-chain. The caching relies on a conventional database, where we use Elasticsearch [44].

Market

The data marketplace [45] queries the metadata store (Aquarius) and displays the data assets retrieved. It lets consumers and publishers interact with the permissionless blockchain layer by leveraging a Web3 wallet like Metamask [23]. The publication and consumption of data assets are enabled by interacting with the respective middleware libraries. The market itself is built on Ocean Protocol JavaScript libraries [46] and React hooks [47].

6.1.2 Data provider

If a data provider wants to sell data assets (using default smart contracts and any public marketplace) to be accessed by download or stream, he does not need to deploy or run anything. He just provides the URL to the data asset, which will be encrypted and stored by Ocean Provider [48]. The keys are held by Ocean Provider too, which is maintained by the entity who runs the marketplace.

In our case, because the access type is restricted to ‘compute only’, the data provider needs to deploy containerized applications where the remote algorithm is executed. This makes him a ‘compute provider’ as well. To keep maximum control over the data and to reduce the risk of data breaches, it is recommended for data providers to take over both roles.

The first software container comprises the Operator Service [49], which manages the compute-to-data workflow, communicates with the data provider, and performs the computation. The second container holds the Operator engine [50], which is a backend service that orchestrates and manages the compute infrastructure.

6.1.3 Data consumer

The data consumer only needs a browser with a Web3 wallet like Metamask [23] to browse and consume datasets.

6.2 Prototype deployment

Each part is set up on its own virtual machine (VM). The virtual machines are connected to the internet, while direct connectivity between them is blocked.

The VM specifications are in all cases:

- Ubuntu 20.04.1 LTS on VMware Hypervisor
- 3GB RAM (+3GB for the infrastructure VM)
- 60 GB HDD
- Internet connection, no local connectivity

6.2.1 VM1 - Infrastructure

Blockchain and smart contracts

A first iteration builds on top of the existing smart contracts by the Ocean Protocol Foundation on Rinkeby test-net. Rinkeby and Ropsten Ethereum test-nets can be used without spending money on gas fees, so they are suitable to test blockchain applications extensively before shifting them to production. Ether, the currency to pay for gas fees on the Ethereum network, can be obtained for free to be used on test-nets by providing an Ethereum address to a faucet [51]. The same is true for the native Ocean token on Ethereum test-nets [52], which will be used for testing. Transactions can be explored and traced using an Ethereum-compatible blockchain explorer like Etherscan [53].

In a second iteration and to allow the data provider to gain maximum control over his data to be sold, we deployed our own contracts to Ethereum. Holding the private keys to the contracts mitigates some risks for the data provider that is described in detail in chapter 7.1. Contracts were deployed using the respective utility available in Ocean Protocol's Github repository [54]. As a prerequisite, it is necessary to obtain an Infura[6] ID [55] and to generate a private/public Ethereum key pair.

Metadata

The metadata component 'Aquarius' [43] was installed and configured to query metadata from the respective contracts. An Elasticsearch database instance was set up and connected to the metadata cache. Aquarius regularly queries the metadata smart contracts, extracts the metadata, and writes it to the Elasticsearch database. Because blockchain interactions are slow, it only searches in the local database when queried for metadata.

[6] Infura is a blockchain Infrastructure-as-a-Service (IaaS) provider that simplifies the access to Ethereum data. Instead of running a dedicated Ethereum node to monitor network transactions, it is possible to query the Infura API for relevant transactions. This allows running lightweight applications on the Ethereum network without spending many resources to monitor the blockchain layer, which is suitable for a prototype implementation. To host a data marketplace in a production environment, it can be considered to deploy a dedicated Ethereum node.

Market
The Ocean Protocol reference marketplace [45] was forked and customized. It interacts with the default contracts on Ethereum main- and test-net. In this particular configuration, the market can be accessed with a browser on port 8000. When browsing for datasets, the market queries the metadata cache on port 5000, which fetches its data from the database at port 9800. Writing operations are carried out through the wallet connected to the browser.

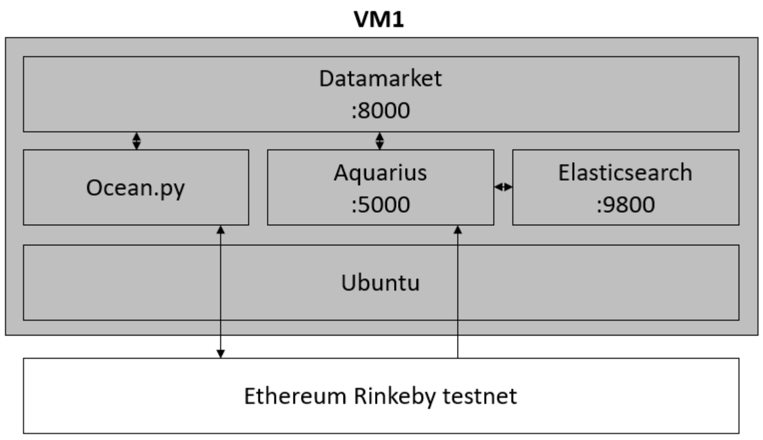
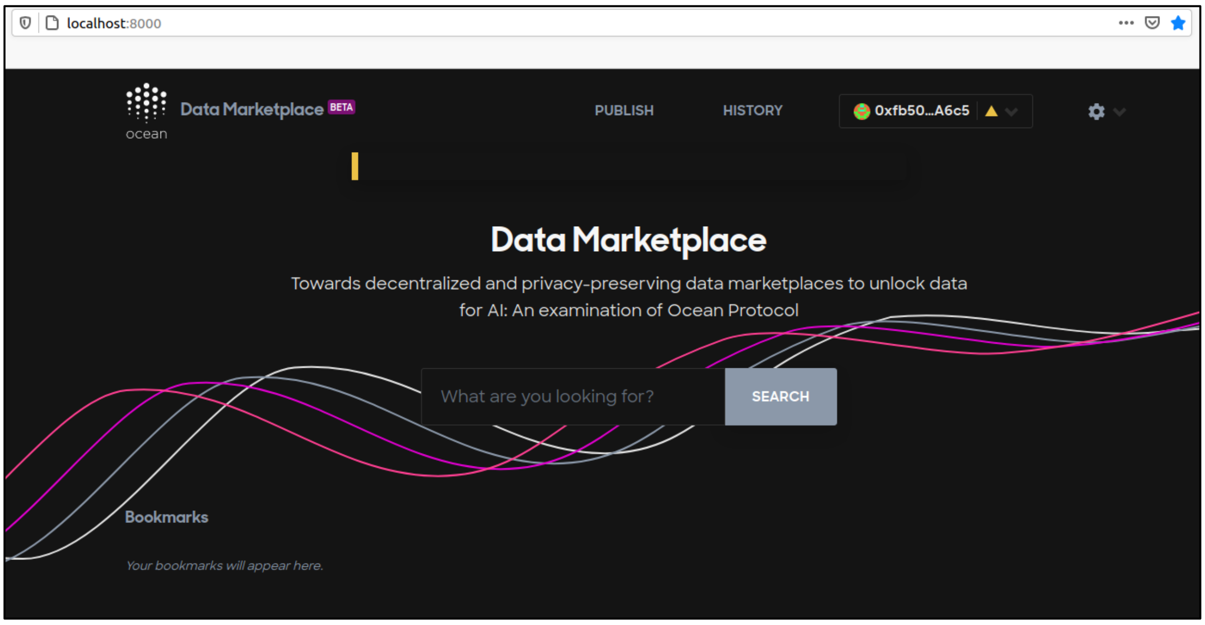


Figure 9: VM1 - Infrastructure



Data Marketplace

Towards decentralized and privacy-preserving data marketplaces to unlock data for AI: An examination of Ocean Protocol

What are you looking for?

Bookmarks

Your bookmarks will appear here.

Figure 8: Ocean Market web interface

6.2.2 VM2 – Data provider

On the provider's side, the Ocean Operator Engine [50] and Operator Service [49] were deployed. Both use a Kubernetes cluster. In this case, Kubernetes Minicube [56] was used in combination with kubectl [57], which require Docker [58] and Docker Compose [59].

The data (and compute) provider need to deploy these containerized applications, provide sufficient resources in accordance with the expected compute jobs and set up read-only access to the storage holding the data. Afterwards, he needs to specify the runtimes (JavaScript, Python) that should be available to compute consumers.

Furthermore, read access to the data needs to be given to the Docker containers. Depending on where the containers are deployed, read-only access to the storage can be configured on the file system level. It must be noted that setting up a secure compute-to-data deployment is not trivial and mostly relies on the security of the underlying infrastructure. The data provider needs to

- ensure the security of the infrastructure where the containers are running
- ensure secure connectivity to the data storage to allow read-only access
- allow (but restrict) internet connectivity to the containers to receive the algorithm to execute
- check and possibly modify or deny the results if they contain sensitive information (this is especially hard and will be discussed further in Section 7.1)
- monitor the whole system

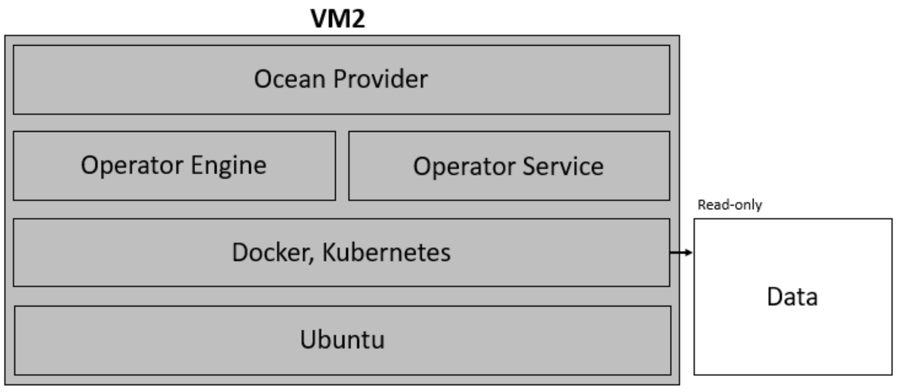


Figure 10: VM2 - Data Provider

6.2.3 VM3 – Data consumer

The system is equipped with a regular web browser and Ethereum-enabled wallet, like Firefox with a Metamask wallet plugin [60].

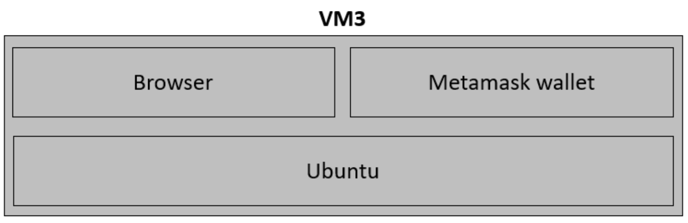
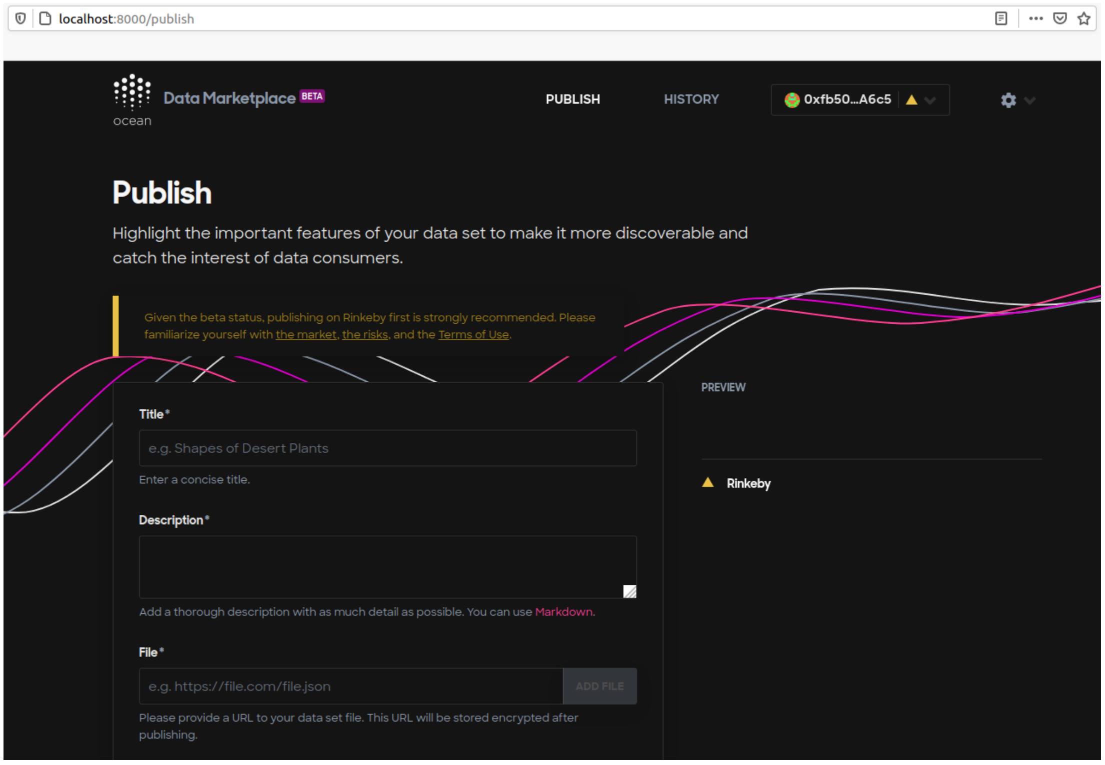


Figure 11: VM3 - Data Consumer

6.3 Prototype demo

To publish a data asset, data owners navigate to the publish tab in the marketplace and follow the steps described in Section 5.7.1.



Publish

Highlight the important features of your data set to make it more discoverable and catch the interest of data consumers.

Given the beta status, publishing on Rinkeby first is strongly recommended. Please familiarize yourself with the market, the risks, and the Terms of Use.

Title*

e.g. Shapes of Desert Plants

Enter a concise title.

Description*

Add a thorough description with as much detail as possible. You can use Markdown.

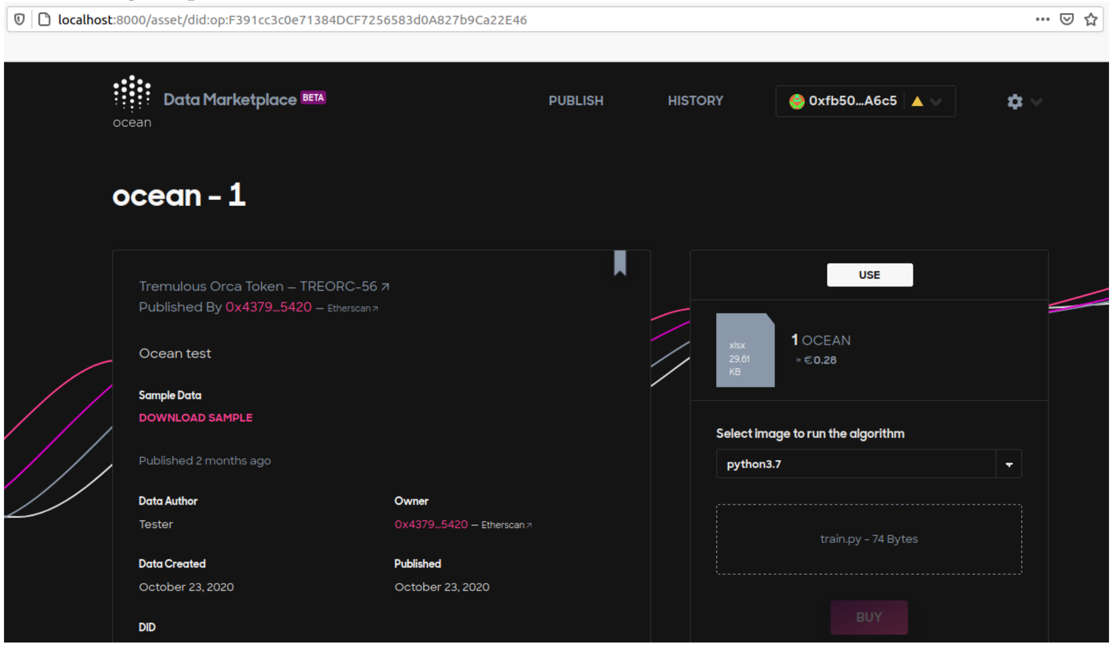
File*

e.g. https://file.com/file.json

Please provide a URL to your data set file. This URL will be stored encrypted after publishing.

Figure 12: Web Interface - Publish

To consume a data asset, consumers can browse the market and follow the steps in Section 5.7.2. In this prototype deployment, they can provide an algorithm implemented in Python or JavaScript, by dropping it into the "use" tab, visible on the right. In order to showcase the use case, it will be executed remotely without further checking. Note that this can be easily exploited by an attacker. Hence, the approach is only appropriate for demonstration purposes in a closed environment or on non-critical data. Data publishers should restrict what algorithms can be executed on their data. The risks are discussed further in the following chapter.



ocean - 1

Tremulous Orca Token — TREORC-56 ↗

Published By 0x4379...5420 — Etherscan ↗

Ocean test

Sample Data

Published 2 months ago

Data Author: Tester

Owner: 0x4379...5420 — Etherscan ↗

Data Created: October 23, 2020

Published: October 23, 2020

DID

1 OCEAN ≈ €0.28

Select image to run the algorithm: python3.7

train.py - 74 Bytes

Figure 13: Web Interface - Consume

7. Evaluation

In comparison to the state-of-the-art data marketplaces presented in Section 2.2, Ocean Protocol offers unique advantages, that result from the combination of blockchain, ‘compute-to-data’ and decentralized computing.

Full control over data assets
With compute-to-data, data owners never give away their data. Because of the decentralized access control powered by smart contracts, centralized data repositories which are in control of the marketplace operator, as used by competing projects, are dismissed. The data owner retains full control.

Token-based access control
Data tokens represent access rights to a data asset. Data tokens can be held, transferred, exchanged and traded with any Ethereum compatible software or device. Following that, with Ocean Protocol, it becomes possible to transfer, exchange and trade data access rights on a decentralized and permissionless network. While fungible (ERC-20) tokens grant equal rights to access the data, non-fungible tokens (ERC-721) enable individual access control. Using Ocean Protocol, holding and transferring data access rights can ultimately be reduced to hold a private key of an Ethereum account.

Efficiency
Data owners can sell data without needing to move it. This favourable for large datasets that are expensive or slow to transfer.

Data auditability
Because data access is tied to data tokens, data auditability and provenance is enabled by design. Any transaction metadata, from registration, sales, transfer, access to remote computation and results are logged on an immutable ledger. Therefore, blockchain explorers become data audit trail explorers. When using Ocean Protocol for data exchange, companies, auditors, and regulators can rely on a tamper-proof source of truth.

Explainability
Data auditability closely relates to the property of ‘explainability’. Explainable AI refers to a principle in AI application, that results can be understood and comprehended by humans. It is the opposite of the black-box approach in machine learning, where researchers cannot explain how and why an AI came to a specific decision.

GDPR mandates the explainability of models. “The data subject shall have the right not to be subject to a decision based solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly significantly affects him or her.” “The data controller shall implement suitable measures to safeguard the data subject's rights and freedoms and legitimate interests, at least the *right to obtain human intervention* on the part of the controller, to express his or her point of view and to contest the decision.” [GDPR Article 22.1, 3] “The data subject shall have the right to obtain from the controller confirmation as to whether or not personal data concerning him or her are being processed, and, where that is the case, access to the personal data and the following information: […] the existence of automated decision-making, including profiling, referred to in Article 22 […], at least in those cases, *meaningful information about the logic involved*, as well as the significance and the envisaged consequences of such processing for the data subject.” [GDPR Article 15.1h] Both human intervention in automated decision making and meaningful information about the logic involved is only possible if a prediction model is explainable.

Ocean Protocol improves AI explainability because, with compute-to-data, a detailed tamper-proof audit trail is available by design. It includes not only which datasets were used for training of a model, but also the algorithm, timestamp and final state (successful, error, cancelled) of the computation. It promotes trust by ensuring that algorithms were correctly executed so that AI developers can be confident with the resulting models.

Data flow between jurisdictions

GDPR enforces strict rules when data containing PII should be transferred outside the European Economic Area. "Any transfer of personal data which are undergoing processing or are intended for processing after transfer to a third country or to an international organisation shall take place only if […] the conditions laid down in this Chapter are complied with by the Controller and processor." [GDPR Art. 44] "A transfer of personal data to a third country or an international organisation may take place where the Commission has decided that the third country, a territory or one or more specified sectors within that third country, or the international organisation in question ensures an adequate level of protection. Such a transfer shall not require any specific authorisation." [GDPR Art. 45] Consequently, personal data cannot be transferred to non-EEA entities, unless there is an agreement issued by the EU Commission that testifies an adequate level of protection.

With data marketplaces on Ocean Protocol, this is of no concern, as with compute-to-data, the data never leaves the premises of the seller. Implied that algorithms do not leak PII or the underlying data, compute access can be sold globally while being GDPR compliant. The physical location of a marketplace built on Ocean Protocol is not relevant either.

7.1 Risks

As described before, Ocean Protocol offers GDPR compliance by design. It is permissionless and operates in a trustless environment. On the one hand, the design choice enables a diverse data supply, equalizes data access and ultimately strengthens data sovereignty. By using a blockchain, many attacks relevant to centralized systems are prevented by design. Hence, many risks applicable to traditional businesses are mitigated by design.

However, the design choice introduces novel risks because anyone can participate in Ocean Protocol and trade data assets, regardless of their reputability or intent. From a business perspective, the setting is uncommon because traditional business models are often based on trust. If a participant is untrusted, at least a trusted entity in the middle manages identities, authentication, and payment. In traditional web2.0 business models like marketplaces, trust is guaranteed by

- Payment providers: Trust in consumers is often ensured by payment providers (PayPal, Amazon|Apple|Google – Pay), banks, postal services and others, verifying identity (KYC, ‘know-your-customer’), creditworthiness and confirming delivery.

- Certificate Authorities: Trust in the providers' identity is ensured by certificate authorities (CA). A certificate authority is a trusted entity that attests providers' identity by issuing digital certificates that certify the ownership of a public key.

Over the last decade, businesses have adapted to web2.0 risks and established processes to monitor and manage risks. As web3.0 protocols, applications, and business models arise, many established processes are likely to become obsolete while novel risks are introduced simultaneously. In order to continually ensure the confidentiality, integrity and availability of company resources, it is necessary to adapt the risk management when deploying web3.0 solutions like Ocean Protocol.

In general, it must be noted that registering sensitive data on Ocean Protocol to be consumed via compute-to-data is a trade-off between risk and reward. It adds a non-zero probability to the risk of data leakage, potentially resulting in financial or reputational loss because it requires additional components, complexity and connectivity to be added to the internal infrastructure. Therefore, this section is dedicated to an Ocean Protocol risk assessment. Risks are identified and evaluated from a business perspective concerning their severity towards security and privacy, ordered from most to least critical.

7.1.1 Malicious algorithms

Risk
With compute-to-data, it is essential that algorithms do not act maliciously. Algorithms returning the data itself, instead of sufficiently aggregating it, defeat the purpose of compute-to-data and pose a critical risk. Given the trustless environment, anyone can buy compute access to data assets. Therefore, the selection, vetting and approval of algorithms need to be managed and restricted effectively. According to GDPR, the data owner is ultimately responsible for ensuring the confidentiality of PII, so he is responsible for governing which algorithms to trust, no matter who is operating the compute container. In addition to losing the data, a publisher faces a financial risk because GDPR specifies fines up to 4% of a company's annual turnover when losing PII at scale.

In a practical attack, an attacker could provide an algorithm that echoes the data. More sophisticated attacks could include code obfuscation or rogue software library imports to disguise their intent.

Severity
The resulting risk of malicious algorithms is critical. It can possibly lead to uncontrolled leakage of sensitive data. Consequently, financial loss (GDPR fines, compensation) and reputational loss are to be expected.

Mitigation strategy:

The execution of algorithms must be restricted effectively
Algorithms must be inspected, vetted and approved before execution. A narrow implementation of vetted algorithms on the Ocean Protocol reference marketplace (GUI) is currently in development by the Ocean Protocol Foundation. Data providers will be able to whitelist and only allow trusted algorithms for remote computation.

However, the functionality is not available as of January 2021. To mitigate the risk, the publisher can restrict the supported algorithms to pre-published known (and vetted) ones. He or she needs to inspect the algorithm and register it as a data assets itself to ensure its integrity. Following that, the algorithm can be allowed for compute-to-data by referencing its DID inside the compute Service Execution Agreement (SEA). Once registered as a trusted algorithm, it can be included in trusted repositories and used by anyone. This workaround is feasible for common use cases and simple computations like in federated analytics. However, the profound vetting of more complicated algorithms remains a challenge. A professional software audit is recommended when dealing with complex algorithms on critical data assets.

Output checking
Before returning, the computation result needs to be inspected concerning data leakage. Output checking requires human experts and manual labour. It cannot be automated effectively as of today [61] [62]. Depending on the complexity of algorithms, the sensitivity of the data and the size of the dataset, the task of manual output checking can occupy a lot of resources.

Residual risk:
Very low, if algorithms are vetted, audited and approved by data owners, then registered as an immutable data asset on the blockchain and output checking is done by human experts.

7.1.2 Data asset integrity

Risk
The integrity of data assets is guaranteed by hashing the asset, storing its hash in the DDO and referencing the DDO with an identifier (DID). Data integrity is critical when trusted algorithms to be used with compute-to-data are registered on-chain, like proposed before. It is essential that the algorithms cannot be altered. However, Ocean Protocol does not *enforce* data integrity. This is because data providers can continuously update their data assets, which is desired for assets like weather and financial data, which can be updated daily or hourly without always interacting with on-chain metadata. If an integrity check is required, the data provider is responsible for providing a checksum to be written into the DDO and stored on-chain.

In case the checksum is missing, an attacker might be able to alter the data asset. Because the integrity cannot be checked automatically, there exists a risk that a malicious compute-to-data algorithm hides behind a trusted DID and DDO and is executed on sensitive data.

Severity
The resulting risk of malicious algorithms, as a consequence of missing integrity checks of data assets, is critical. It can possibly lead to uncontrolled leakage of sensitive data. Consequently, financial loss (GDPR fines, compensation) and reputational loss are to be expected.

Mitigation strategy
The integrity of a data asset can be guaranteed by setting the ‘integrity’ DDO file attribute.

The data provider is responsible for taking care of the integrity check when first registering the data asset. The checksum is then stored on-chain as part of the DDO.
Moreover, when consuming a compute service with compute-to-data, the integrity check needs to be performed again once the algorithm arrives at the compute provider.

Residual risk
None, if a checksum was provided at the time of publishing and integrity is rechecked before data consumption.

7.1.3 Data token leaks, theft and misuse

Risk

Anyone who owns a data token can access the corresponding data assets. Because of the permissionless approach, data tokens can be traded anywhere. If tokens are stolen, leaked, sold on a black market, or simply sent to a wrong address, the new token holder can consume the data asset.
Depending on the data and related regulation (i.e. health data, personal data), this circumstance can pose a critical risk. For those use cases, it is essential to be able to restrict access to authorized groups; for example, only registered medical personnel is allowed to access patient data.

Severity

The resulting risk of data token leaks, theft and misuse can be high to critical, depending on the grade of data sensitivity. When compute-to-data access tokens are affected, it can possibly lead to uncontrolled leakage of sensitive data. Consequently, financial loss (GDPR fines, compensation) and reputational loss are to be expected.

Mitigation strategy

To mitigate the risk, group-restricted access to marketplaces is needed to limit who can consume data. The restriction might be needed at the level of organizations, companies, departments or roles.

Possible ways to restrict access to authorized groups are:

(1) Restrict at ERC20 contracts level
(2) restrict access to the marketplace
(3) restrict the ability to buy in the marketplace
(4) restrict at the point of consumption.

Option 2 and 3 are insecure because they do not prevent misuse of leaked or stolen data tokens.

Restrictions in the ERC20 data token smart contract can restrict the transfer and use of data tokens to authorized groups. The restriction can be implemented by referencing authorized addresses in the immutable contract. The contract only accepts transactions from these addresses and to transfer a data token, authorized users need to sign the respective transaction with their private key. Given the smart contract properties discussed before, the solution offers the highest security of the four approaches. On the other hand, it requires a custom implementation which modifies the data token contract's transfer(), approve() and transferFrom() functions. This needs to be done for every contract by revising the contract template, which is always instantiated at the time of token generation through an ERC-1167 proxy contract. The change requires a security audit. It might be appropriate for organizations who desire maximum control, have the resources to implement and maintain custom contracts and whose data is sensitive or regulations are strict enough to justify the effort.

Option 4 restricts at the point of consumption. This could be implemented by introducing credentials that are checked by Ocean Provider. A possibility is to use Verifiable Credentials (VC), where the issuing authority signs an attestation that a DID belongs to a credential. Using this approach significantly lowers the consequences and the resulting risk of data token leaks or theft because tokens can only be consumed if the consumer holds a Verifiable Credential.

All in all, restrictions at ERC20 contract level can be considered most secure but require a custom implementation. It could be appropriate for specific use cases where the criticality of data assets justifies the expenditure. For standard use cases and depending on the criticality of data assets, a restriction can be sufficient at the point of consumption. It is less complex to develop, implement and maintain and effectively restricts access to authorized groups if the VC is not compromised.

Residual risk

None, if restrictions at contract level are correctly implemented.
Very low, if data assets are bound to Verifiable Credentials.

7.1.4 Model leaks, overfitting, unintended memorization

Risk With compute-to-data, it is essential that algorithms do not act maliciously. However, algorithms do not necessarily be intentionally malicious to pose a risk. In some cases, algorithms can exfiltrate PII without the data consumers intent. Possible ways are model leaks, training set inferences, overfitting or unintended memorization. - **Model leaks:** Federated learning and related techniques might leak unintended information about the data providers datasets. Inference attacks can exploit this leakage. [63] “A curious party can infer the distribution of sensitive attributes in other parties’ data with high accuracy. This raises concerns regarding the confidentiality of properties pertaining to the whole dataset as opposed to individual data records. […] Leakage occurs even if the sensitive attribute is not included in the training data.” [64] - **Overfitting:** With overfitting, a model corresponds too closely or exactly to the training set. It is also referred to as ‘overtraining’. The model unknowingly extracted and remembered some of the training examples instead of learning to generalize [65]. If the model was trained on PII, it could unknowingly exfiltrate personal data. - **Unintended memorization** appeals to some types of machine learning models, that “have a tendency to memorize rare or unique sentences in the training data.” […] The prediction model can leak information about the underlying training data in unexpected ways. […] “Since useful models are often trained on sensitive data, to ensure the privacy of the training data, it is critical to identify and mitigate such unintended memorization” [66] The risks are fundamental to federated learning and not specific to Ocean Protocol. Still, they can be relevant when remotely training machine learning models with compute-to-data.
Severity The resulting risk of model leaks, overfitting and unintended memorization, can be high to critical, depending on the sensitivity of the training data. When training on personal data, it can possibly lead to uncontrolled leakage of PII. If the leak affects a large number of individuals, financial loss (GDPR fines, compensation) and reputational loss are to be expected.
Mitigation strategy Mitigation is a challenging task because the risks are fundamental to federated learning and not specific to Ocean Protocol or dependent on a particular implementation. Moreover, as risks might even be realized without the presence of an attacker, as an unintended by-product of the computation, detection and prevention is challenging.
Residual risk The unknown exfiltration of PII remains a possibly critical risk, that demands further consideration and research. Depending on data sensitivity (especially “special categories of personal data” according to GDPR Art. 9) the risk can be critical. Viewed on a timeline, the risk is minimally lower in the short-term because of the absence of an immediate attacker.

7.1.5 Model poisoning

Risk
With model poisoning, a malicious data provider aims to attack a data consumer by injecting bad data into the training set, so that the machine learning model learns something unintended. Poisoning attacks can be differentiated in targeted and untargeted attacks. [67]

- Untargeted attacks have the goal to corrupt a model by injecting bad data, such that the accuracy drops until the model becomes useless.
- Targeted attacks, also known as **backdoor attacks**, aim to preserve the models' accuracy except for one exception, which is the backdoor. “The objective of the attacker is to create a backdoor that allows the input instances created by the attacker using the backdoor key to be predicted as a target label of the attacker’s choice. For example, performing backdoor attacks against face recognition systems enables the attacker to impersonate another person. Thus the attacker can mislead the authentication system into identifying him as a person that has access to a building or a device so that the attacker can get into a place or a system that he originally cannot access.“ [68]

Model poisoning attacks are a fundamental risk of federated learning and not specific to Ocean Protocol. However, because of the trustless environment, everyone can provide training data, and Ocean Protocol does not provide any protection. It does not check the integrity of data assets offered for remote computation with compute-to-data. Even if such a feature existed, an attacker could easily circumvent it by registering malicious datasets in the first place. Furthermore, a malicious data provider does not necessarily need to provide bad data. He also could just replace the model with its own backdoored one before returning it. [69]

Severity
Untargeted attacks are easy to detect because the model does not work as intended. They mainly result in a financial loss because they waste computational and data resources.

However, targeted attacks are hard to detect, because the model provides the intended accuracy in all cases, except for the backdoor, which is only known to the attacker. The attack can remain unrecognized for a long time. Besides financial loss, an attack could lead to safety issues and reputation loss.

Mitigation strategy
Existing mitigation strategies against backdoor attacks require examination of the training data or full control of the training procedure [70, 71]. Both are not possible with compute-to-data.
To mitigate the risk and when training models where safety is critical, data consumers should rely on trusted data providers.

Technically, the most common defence strategy to model poisoning is outlier/anomaly detection. Because adversarial input is often very different from genuine input, it can be detected in theory. Nevertheless, in practice, the approach is problematic. If the poisoning already happens during the first iterations, or the adversarial inputs are very similar to genuine input, outlier detection is defeated.

Residual risk
Low, if data consumers only rely on trusted data providers.

7.1.6 Profiling

Risk
Full transparency is a fundamental property of permissionless blockchains. Hence, smart contracts, token transfers and applications based on blockchain inherit those properties. While transparency has advantages with respect to auditability and explainability, it introduces risks. An organization's transactions can be monitored and analyzed, enabling profiling by competitors. Applied to Ocean Protocol, data token transfers which relate to data sales and consumes are publicly recorded on-chain. Moreover, the transactions can be matched to specific data assets and consumes. As compute-to-data is mainly used to train machine learning models, competitors might infer information about internal research and development projects.

Furthermore, it has to be noted that Ethereum smart contracts, their state and logic, and inputs and outputs are public and logged on-chain.

Severity
The resulting risk of blockchain profiling can vary from none to medium risk, depending on the use case. It does affect metadata, from which competitors might infer information about internal research and development projects.

Mitigation strategy
Full transparency is a fundamental property of Ocean Protocol. Market participants can try to obfuscate their identity and actions, but the transactions will always be recorded on a public ledger.
A basic workaround to hinder profiling is to generate a new private/public key-pair for a transaction like each data consume. However, the shuffling approach can be considered as 'security by obscurity' and only raises the outlay of profiling but does not mitigate the risk, especially with respect to blockchain forensics.

When sharing data within a branch or group of companies and confidentiality on the transaction layer is needed, deployment on a private blockchain ('consortia blockchain') is possible. It could be bridged to the main Ethereum network to allow for interoperability.

Besides, there exists technology to ensure confidentiality of smart contract input, logic and states. One possibility is "Secret Network" [72], which can be combined with Ocean Protocol to enable secret on-chain computation and storage to be used for metadata, SEAs and similar functionality. Secret Network (former known as "Enigma" [73]) aims to encrypt smart contract input and output data as well as state, ensuring confidentiality, also towards node operators executing contracts. The feature is implemented by encapsulating smart contracts into Trusted Execution Environments (TEE). More information is available here [74].

Residual risk
The residual risk of blockchain profiling and forensics, when applying obfuscation methods, can vary from none to medium risk, depending on the use case.
It can possibly be lowered in future releases that leverage "Secret Network" to conceal smart contract input, output and state.

7.1.7 URL disclosure

Risk URLs to data assets are encrypted before they are stored on-chain. The Ocean Provider component carries out the encryption. Hence, Ocean Provider also has access to plain text URLs. There exists a risk that a malicious Ocean Provider operator misuses his position to gain access to data asset URLs. An attacker who operates his own marketplace *and* Ocean Provider can consume data assets (download, stream) without interacting with smart contracts because the URL is known. The attack does not apply to compute-to-data, as a valid SEA is needed to initiate remote compute jobs. In general, operating Ocean Provider or relying on a third-party Provider is a trade-off between trustlessness and convenience. Data publishers can either have a fully trustless operation (by deploying and operating their own Ocean Provider) or convenience (by just publishing on a third-party Provider), but not both at the same time.
Severity The risk is low because an attacker only gets access to data that is intended to be sold and accessed by download or stream. Consequently, it cannot contain PII. The attack circumvents the payment and is limited to the entity who controls Ocean Provider. In the worst case, the attacker gets access to data intended to be shared but does not need to pay. The attack results in a financial loss, but no risk of data leakage.
Mitigation strategy For now, organizations that want to sell valuable data at scale should consider deploying their own marketplace and Ocean Provider. The risk does not apply when Ocean Provider is not operated by a third party. If deploying a dedicated market is not an option, data providers should sell their data on trusted marketplaces, like the Ocean Protocol Foundations reference marketplace or other marketplaces operated by reputable entities. In the future, Secret Network can be leveraged to encrypt URLs. It offers a practical solution to encrypt URLs in a decentralized way without relying on Ocean Provider. The URL can be provided as an encrypted input into a secret smart contract, stored in its state, and the URL can only be decrypted by the data token of the corresponding data asset. This eliminates the need for a trusted entity. The solution is currently under development [75].
Residual risk None, if running a self-owned marketplace and Ocean Provider. Very low, if relying on trusted third-party marketplaces.

7.1.8 Smart contract upgradeability

Risk Smart contracts are immutable by default. However, they can be designed to be upgradeable. Upgradeable smart contracts pose a risk because a malicious contract owner could abuse his power to upgrade a contract. The contracts could grant access to data assets on the malicious owners' conditions, for example, without requiring payment or data tokens. Technically, updates can be made upgradeable by using proxy contracts. Relevant contracts within Ocean Protocol (metadata, fixed-rate exchange, liquidity pool contracts) are immutable. However, an attacker could deploy his own market based on proxy contracts, regularly operate the market for some time and upgrade the contracts once enough data assets are registered on his market. In this case, the risk exists that the attacker and other market participants can consume data assets (download, stream or compute) without needing to pay. The proxy setting should not be confused with the ERC1167 proxy contracts used for data token generation. In contrast, they increase security because the contract does nothing but proxying to its hard-coded immutable reference.
Severity The risk is low to medium. In case of a successful attack, the attacker and any market participant get access to data assets intended to be sold and accessed by download or stream. Hence, they do not contain PII. The attack circumvents the payment or data token transfer. In the worst case, data that is intended to be shared can be accessed for free. The attack results in financial loss, but no risk of data leakage.
Mitigation strategy Before publishing data on third-party marketplaces, data providers should check if smart contracts were designed to be upgradeable. If they are not, the risk does not apply.
Residual risk None.

7.2 Usability, scalability and costs

Scalability

Ocean Protocol is currently only deployed on the Ethereum blockchain. Hence, it is limited to Ethereum's capabilities. Ethereum in its current state scales to 15 transactions per second [76] and the transaction capacity is shared between all applications building on top. Moreover, as transactions are chained since its inception, every node needs to store the full blockchain history. The ledger had reached a size of 195GB as of late 2020 since its genesis block in July 2015. [77]

Cost

Storage on the Ethereum blockchain is expensive. In late 2020, persistent storage of one megabyte costs several thousand Euros.[7] To lower gas expenditures, Ocean Protocol stores minimal data on-chain. Storage and transaction capacity are kept to a minimum by leveraging proxy contracts, off-chain metadata for non-DDO attributes and metadata caching. Moreover, gas fees are volatile, as pricing in Ethereum depends on the utilization of the network. Prices usually fluctuate with user activity. Average prices to publish a data asset have been between 20 and 30 EUR in late 2020. To give a practical example, the publication of a data asset on the Ethereum main-net[8] as of late 2020 resulted in gas fees of about 30 EUR. As described before, it is paid once by the data publisher. A consume of the same data asset did cost the buyer 2.30 EUR.[9]

Transaction	**Price in EUR**
Create token	6.84 [10]
Publish metadata	3.53 [11]
Create liquidity pool:	3.81 [12]
Approve	1.10 [13]
Setup liquidity pool	14.70 [14]
Total	**29.98**

Usability

While the Ethereum blockchain is sufficient to demonstrate the concept of decentralized data sharing, it is presumably not capable enough to power a larger decentralized data economy with its throughout of 15 transactions per second. Besides Ocean Protocol, Ethereum currently hosts another 1700 active decentralized applications. [78] In addition, the publication costs of up to 30 EUR per data asset are ineligible for micro-assets like IoT data and exclude many use cases relevant to the data sovereignty of individuals, especially in countries of low income. For large and valuable data assets, the throughput is sufficient for now. However, Ocean Protocol does consider itself blockchain agnostic. It can be deployed to any chain supporting ERC20 contracts. Despite their name, ERC20 contracts became a de-facto smart contract standard and are now being implemented on many distributed ledgers. The goal for Ocean Protocol is to become "multi-chain" and to support many blockchains with respect to different use cases. [79] Data assets that require the security of Ethereum because of their value or sensitivity can reside on Ethereum, while use cases like IoT streaming data can in the future migrate to a chain with lower fees and level of security but more throughput.

[7] To store a 256 bit word on the Ethereum blockchain costs 20.000 gas fees [33]. A megabyte is thus 655,360,000 gas. The average gas price as of late 2020 is about 20 gwei (0.00000002 ETH), so a megabyte of storage costs about 13 ETH which is circa 6.000 EUR (at 470 EUR per ETH in December 2020). It must be noted that prices are volatile. Gas prices change continually depending on blockchain utilization. ETH prices are subject to market volatility.
[8] https://market.oceanprotocol.com/asset/did:op:7Bce67697eD2858d0683c631DdE7Af823b7eea38
[9] https://etherscan.io/tx/0x9625709e4bf4f51991a9e9bc3e95c9338e942c0ce075e9323b75603e65f2c4af
[10] https://etherscan.io/tx/0x8471e70e590fc2863ff9c403371658198f89e991e1220592460f36fd7ec03092
[11] https://etherscan.io/tx/0xf20705123c2907c5f07981a7a8f78f018c0e04d4be73fcafc30866c2eec48f71
[12] https://etherscan.io/tx/0x3290f70b3cc6e7889a7e3be009ae5a1d7c54823ac4114ff18d45bd456665e513
[13] https://etherscan.io/tx/0x757a1eea326d061475c32ed4d4dc5074928bf3f902caf2ec5267fc3658a69b3e
[14] https://etherscan.io/tx/0x23ad38f4a8dfc0598c659044404e328124835868c8309b61357835fda4526544

7.3 Limitations

Given the risks and usability/scalability issues for specific use cases, Ocean Protocol has limitations that prevent its unlimited large-scale use today. Compared to the state-of-the-art data marketplaces presented in Section 2.2, most risks are not unique to Ocean Protocol, but a fundamental issue with privacy-preserving data marketplaces. While the decentralized approach solves many issues the centralized competition faces, it indeed introduces novel difficulties with regard to scaling, cost and (too much) transparency.

All in all, the major limitations of all privacy-preserving data marketplaces are:

- **Malicious algorithms**
  Algorithms returning the data itself, instead of sufficiently aggregating it, pose a critical risk. It can possibly lead to uncontrolled leakage of sensitive data. The issue can be solved if algorithms are vetted, audited and approved by data owners.

- **Output checking**
  Before returning, the results of remote computations need to be inspected concerning data leakage. Output checking requires human experts and manual labour. It cannot be automated effectively as of today, and mistakes can possibly lead to uncontrolled leakage of sensitive data.

- **Model poisoning**
  By injecting bad data into a training set, a malicious data provider can sabotage machine learning models of consumers. Targeted attacks are hard to detect and can remain unrecognized for a long time. Besides financial loss, an attack could lead to safety issues and reputation loss.

Unique to Ocean Protocol are the limitations of

- **Full transparency**
  Blockchain transactions can be monitored and analyzed, enabling profiling by competitors. With Ocean Protocol, data token transfers which relate to data sales and consumes are publicly recorded on-chain. Competitors might be able to infer information about an organization's internal research and development projects.

- **Costs**
  Blockchain storage is expensive. Despite being as lightweight as possible, the publication of data assets costs up to 30 EUR per asset. This does not qualify Ocean Protocol for use cases related to micro-data, IoT and mass exchange of smaller assets.

- **Scalability**
  Ethereum scales to 15 transactions per seconds while hosting another 1700 decentralized applications. The scalability is presumably not sufficient for a mature decentralized data economy.

But, to our best estimate, the remaining problems of Ocean Protocol that prevent mass-usage can be solved with further developed distributed ledger technology, that enables scalability, lower fees, and confidential distributed computation. With the blockchain-agnostic approach, Ocean Protocol can be ported with little effort to whichever distributed ledger offers the best solution in the future.

On the other hand, the state-of-the-art centralized privacy-preserving marketplaces remain fundamentally flawed because of their underlying principle of custodial data repositories, where data owners lose control over their data. They need to move and pseudonymize their assets and are reliant on a third party to keep their data safe if they want to use such a marketplace.

8. Conclusion

Compared to the state-of-the-art data marketplaces presented in Section 2.2, Ocean Protocol is the first technology to enable the possibility of trading and exchanging data and corresponding access rights in an open, permissionless and trustless environment while preserving privacy. The technology theoretically enables individuals and institutions around the globe to participate in a global data economy while complying with privacy regulations. Hence, Ocean Protocol offers the tools to strengthen the data sovereignty of data market participants.

To our best knowledge, it is the first solution which brings the conjunct properties of permissionless-ness, trustlessness, immutability, accountability, token-based access control, token-based non-custodial data ownership, censorship resistance and GDPR-compliance to the world of data sharing.

The deployment of our prototype confirmed the fundamental functionality in practice. We detected no drawbacks that considerably hinder the utilization of Ocean Protocol in practice to build privacy-preserving data sharing solutions on top. Especially when used with large and valuable data assets, Ocean Protocol is a suitable and, compared to the state-of-the-art data marketplaces presented in chapter 2.2, superior solution.

Opportunities
Ocean Protocol with compute-to-data appears to be a promising toolset, especially for data scientists and AI researchers:

- It opens access to data which was previously inaccessible.
- Data owners retain control of their data. It never leaves their premises.
- Data owners can sell data access without moving it, which is convenient and inexpensive when dealing with large datasets.
- It ensures compliance with privacy regulations like GDPR because data containing PII is not moved or copied internally, nor accessed by third parties.
- It promotes trust by ensuring that algorithms were correctly executed so that AI researchers can be confident with the resulting models.
- It ensures auditability and provenance because all transactions are recorded on-chain, forming a tamper-proof audit trail.
- It improves the explainability of models (which is also mandated by GDPR).
- It allows for selling trained AI models or synthetic data obtained by using compute-to-data.

The concept enables privacy-preserving data sharing across organizations. To give an example, this is especially useful when applied to verticals like the health sector, where Ocean Protocol enables the possibility to train machine learning models across patient data residing at different hospitals. Because the data never leaves its location, the administrative and regulative overhead remains low.

Moreover, the decentralized approach enables unique features that centralized marketplaces cannot offer by design:

- Token-based access control on an immutable ledger
- Automated decentralized price discovery for data assets
- Data curation where curators are incentivized to find quality data assets
- Censorship-resistance
- Non-custodial data exchange
- No single point of failure
- Provenance
- Auditability

The security and privacy characteristics of Ocean Protocol mainly depend on the trustfulness of compute-to-data algorithms which is the most crucial property to monitor when used in practice. Because the inspection of computing output remains a fundamental problem, we advise restricting the remote execution by following a whitelist approach to only allow vetted and trusted algorithms. When dealing with complex algorithms, we recommend a professional audit. Afterwards, trusted algorithms can be registered as a data asset on-chain while their integrity is ensured by design.

Furthermore, it must be noted that profiling of transactions is generally possible because of the reliance on a public blockchain layer. It can be contained to a certain extent by shuffling addresses and moving sensitive interactions onto "Secret Network" based smart contracts. For highly sensitive data, we advise that the compute provider himself/herself publish and stores his/her trusted algorithms to prevent attacks that go beyond the functional capabilities of Ocean Protocol.

When following said recommendations, we conclude that Ocean Protocol enables secure privacy-preserving data sharing of large and valuable data assets at scale and is a suitable tool to sell latent but sensitive data assets.

However, it must be noted that selling compute access adds a small but non-zero probability to the risk of data leakage because it requires additional components, complexity and connectivity to be added to the internal infrastructure. If the resulting risk of allowing connectivity from internal storage systems to DMZ is not acceptable, data can be replicated to DMZ. If both options are considered inappropriate, compute-to-data cannot be used because the compute cluster (which needs to have read access to the private data) needs to communicate with Ocean Operator components which relay information to and from the internet. When in doubt, we recommend a risk assessment, including regulatory risk, to evaluate if the risk-reward-tradeoff is justified.

9. Future research

Future work and research are mainly related to the risk of remote computation. The appropriate vetting, audit and standardization of trusted algorithms need to be analyzed with the overall goal to automate as much as possible while keeping the risks to an acceptable level. Because manual output checking remains as a fundamental problem and human task within any privacy-preserving data exchange, minor additional human work with respect to the quality control of algorithms can be accepted.

As the main technical limitations of the Ocean Protocol toolset today are scalability, cost, and being overly transparent, those are the most urgent to work on to enable a broad and diverse data economy on top. However, the limitations are resulting from the underlying blockchain layer and not from the Ocean Protocol implementation. Hence, if we consider a permissioned blockchain inappropriate and do not want to develop one ourselves, we must wait for blockchain developments to catch up with real-world requirements. Whenever a scaling solution with lower or no fees appears, Ocean Protocol is ready to be deployed in production and at scale for every type of data asset.

10. Bibliography

References

[1] Ocean Protocol, *Roadmap.* [Online]. Available: https://oceanprotocol.com/technology/roadmap (accessed: Nov. 29 2020).

[2] *BigchainDB GmbH.* [Online]. Available: https://www.bigchaindb.com/ (accessed: Dec. 13 2020).

[3] *Ocean Protocol Foundation.* [Online]. Available: oceanprotocol.com/about (accessed: Dec. 13 2020).

[4] Ocean Protocol, *Github.* [Online]. Available: github.com/oceanprotocol (accessed: Dec. 13 2020).

[5] ISO 25237:2017, *Health informatics - Pseudonymization.* [Online]. Available: https://www.iso.org/standard/63553.html (accessed: Dec. 29 2020).

[6] *Data Republic.* [Online]. Available: https://www.datarepublic.com/ (accessed: Dec. 30 2020).

[7] *OPAL Project.* [Online]. Available: https://www.opalproject.org/ (accessed: Dec. 30 2020).

[8] *X-Road Data Exchange.* [Online]. Available: https://x-road.global/ (accessed: Dec. 30 2020).

[9] *TensorFlow Federated.* [Online]. Available: https://www.tensorflow.org/federated (accessed: Dec. 12 2020).

[10] *OpenMined.* [Online]. Available: https://www.openmined.org/ (accessed: Dec. 12 2020).

[11] F. M. Bencic and I. Podnar Zarko, "Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph," pp. 1569–1570, doi: 10.1109/ICDCS.2018.00171.

[12] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[13] D. Tapscott and A. Tapscott, *Blockchain revolution: How the technology behind bitcoin is changing money, business, and the world*. New York: Portfolio / Penguin, 2016.

[14] Ethereum Foundation, *Introduction to smart contracts.* [Online]. Available: https://ethereum.org/en/developers/docs/smart-contracts/ (accessed: Nov. 17 2020).

[15] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, "Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends," *IEEE Trans. Syst. Man Cybern, Syst.*, vol. 49, no. 11, pp. 2266–2277, 2019, doi: 10.1109/TSMC.2019.2895123.

[16] Ethereum Foundation, *Solidity.* [Online]. Available: https://github.com/ethereum/solidity (accessed: Nov. 17 2020).

[17] Ethereum Foundation, *Solidity 0.7.4 documentation.* [Online]. Available: https://docs.soliditylang.org/en/v0.7.4/ (accessed: Nov. 17 2020).

[18] Ethereum Foundation, "EIP-20: ERC-20 Token Standard," 19 Nov., 2015. https://eips.ethereum.org/EIPS/eip-20 (accessed: Nov. 17 2020).

[19] Ethereum Foundation, "EIP-721: ERC-721 Non-Fungible Token Standard," 24 Jan., 2018. https://eips.ethereum.org/EIPS/eip-721 (accessed: Nov. 17 2020).

[20] National Institute of Standards and Technology, "Digital Signature Standard (DSS)," *U.S. Department of Commerce*, 2013, doi: 10.6028/NIST.FIPS.186-4.

[21] Ocean Protocol, *Data Tokens.* [Online]. Available: oceanprotocol.com/technology/data-tokens (accessed: Oct. 27 2020).

[22] *OAuth 2.0 — OAuth.* [Online]. Available: https://oauth.net/2/ (accessed: Nov. 27 2020).

[23] *MetaMask Wallet.* [Online]. Available: https://metamask.io/ (accessed: Dec. 9 2020).

[24] *MyEtherWallet.* [Online]. Available: myetherwallet.com/ (accessed: Oct. 12 2020).

[25] *Trezor Hardware Wallet.* [Online]. Available: https://trezor.io/ (accessed: Oct. 12 2020).

[26] *Ledger Hardware Wallet.* [Online]. Available: https://www.ledger.com/ (accessed: Oct. 12 2020).

[27] *Uniswap.* [Online]. Available: uniswap.org (accessed: Nov. 18 2020).

[28] Balancer Labs, *Balancer Finance.* [Online]. Available: https://balancer.finance/ (accessed: Nov. 18 2020).

[29] Ocean Protocol, *Ocean Market.* [Online]. Available: github.com/oceanprotocol/market (accessed: Nov. 27 2020).

[30] B. Pon, "Ocean V3 Is Now Live - Ocean Protocol," *Ocean Protocol*, 27 Oct., 2020. https://blog.oceanprotocol.com/ocean-v3-is-now-live-b47c0e73f52a (accessed: Dec. 13 2020).

[31] T. Mcconaghy, "Ocean Protocol V3 Architecture Overview - Ocean Protocol," *Ocean Protocol*, 10 Dec., 2020. https://blog.oceanprotocol.com/ocean-protocol-v3-architecture-overview-9f2fab60f9a7 (accessed: Oct. 26 2020).

[32] Ethereum Foundation, "EIP-1167: Minimal Proxy Contract," 22 Jun., 2018. https://eips.ethereum.org/EIPS/eip-1167 (accessed: Nov. 27 2020).

[33] Dr. Gavin Wood, "Ethereum: A secure decentralised generalised transaction ledger - EIP 150 Revision," [Online]. Available: http://gavwood.com/paper.pdf

[34] Ocean Protocol, *Ocean Protocol Enhancement Proposals - OEPs 8: Assets Metadata Ontology.* [Online]. Available: github.com/oceanprotocol/OEPs/tree/master/8/v0.5 (accessed: Nov. 27 2020).

[35] W3C, *Decentralized Identifiers (DIDs) v1.0.* [Online]. Available: https://w3c.github.io/did-core/ (accessed: Nov. 27 2020).

[36] Ocean Protocol, *Ocean Protocol Enhancement Proposals - OEPs 7: Decentralized Identifiers.* [Online]. Available: github.com/oceanprotocol/OEPs/blob/master/7/v0.3/ (accessed: Nov. 27 2020).

[37] Balancer Labs, *Balancer Finance.* [Online]. Available: https://balancer.finance/ (accessed: Dec. 12 2020).

[38] Ocean Protocol, "Compute-to-Data," oceanprotocol.com/technology/compute-to-data (accessed: Nov. 29 2020).

[39] VMware, *What is a Kubernetes Cluster?* [Online]. Available: https://www.vmware.com/topics/glossary/content/kubernetes-cluster (accessed: Nov. 28 2020).

[40] D. de Jonghe, "Exploring the SEA: Service Execution Agreements - Ocean Protocol," *Ocean Protocol*, 30 Nov., 2018. blog.oceanprotocol.com/exploring-the-sea-service-execution-agreements-65f7523d85e2 (accessed: Dec. 14 2020).

[41] Ocean Protocol, *Ocean Protocol Enhancement Proposals - OEPs 12: Execution of Compute Services.* [Online]. Available: github.com/oceanprotocol/OEPs/tree/master/12 (accessed: Nov. 28 2020).

[42] Ocean Protocol, *Smart contract ontract addresses for Rinkeby, Ropsten, Mainnet* (accessed: Jan. 8 2021).

[43] Ocean Protocol, *Aquarius.* [Online]. Available: https://github.com/oceanprotocol/aquarius (accessed: Jan. 8 2021).

[44] *Elastic Stack: Elasticsearch.* [Online]. Available: https://www.elastic.co/de/elastic-stack (accessed: Nov. 12 2020).

[45] Ocean Protocol, *Ocean Market.* [Online]. Available: github.com/oceanprotocol/market (accessed: Oct. 12 2020).

[46] Ocean Protocol, *ocean.js.* [Online]. Available: https://github.com/oceanprotocol/ocean.js (accessed: Jan. 8 2021).

[47] Ocean Protocol, *React.* [Online]. Available: https://github.com/oceanprotocol/react (accessed: Jan. 8 2021).

[48] Ocean Protocol, *Provider.* [Online]. Available: https://github.com/oceanprotocol/provider (accessed: Jan. 8 2021).

[49] Ocean Protocol, *Operator-Service.* [Online]. Available: https://github.com/oceanprotocol/operator-service (accessed: Jan. 8 2021).

[50] Ocean Protocol, *Operator-Engine.* [Online]. Available: https://github.com/oceanprotocol/operator-engine (accessed: Jan. 8 2021).

[51] *Rinkeby: Authenticated Faucet.* [Online]. Available: https://faucet.rinkeby.io/ (accessed: Oct. 12 2020).

[52] *Ocean Rinkeby Token Faucet.* [Online]. Available: faucet.rinkeby.oceanprotocol.com (accessed: Oct. 12 2020).
[53] *ETH Blockchain Explorer - Etherscan.* [Online]. Available: https://rinkeby.etherscan.io/ (accessed: Nov. 12 2020).
[54] Ocean Protocol, *Contracts.* [Online]. Available: https://github.com/oceanprotocol/contracts (accessed: Jan. 8 2021).
[55] Infura, *Ethereum API.* [Online]. Available: https://infura.io/ (accessed: Oct. 12 2020).
[56] Kubernetes, *Installation of Minikube.* [Online]. Available: https://kubernetes.io/de/docs/tasks/tools/install-minikube/ (accessed: Oct. 12 2020).
[57] Kubernetes, *Install and Set Up kubectl.* [Online]. Available: https://kubernetes.io/docs/tasks/tools/install-kubectl/ (accessed: Oct. 12 2020).
[58] Docker Documentation, *Install Docker Engine.* [Online]. Available: https://docs.docker.com/engine/install/ (accessed: Oct. 12 2020).
[59] Docker Documentation, *Install Docker Compose.* [Online]. Available: https://docs.docker.com/compose/install/ (accessed: Oct. 12 2020).
[60] *MetaMask – Firefox Extension.* [Online]. Available: https://addons.mozilla.org/de/firefox/addon/ether-metamask/ (accessed: Nov. 12 2020).
[61] Steve Bond, Maurice Brandt, Peter-Paul de Wolf - DwB (Data without Boundaries), "Guidelines for Output Checking: Improved Methodologies for Managing Risks of Access to Detailed OS Data," [Online]. Available: https://ec.europa.eu/eurostat/cros/system/files/dwb_standalone-document_output-checking-guidelines.pdf
[62] Safe Data Access Professionals Working Group (SDAP), "Handbook on Statistical Disclosure Control for Outputs," [Online]. Available: https://ukdataservice.ac.uk/media/622521/thf_datareport_aw_web.pdf
[63] Luca Melis and Congzheng Song, Emiliano De Cristofaro, *Exploiting Unintended Feature Leakage in Collaborative Learning.* [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8835269 (accessed: Dec. 30 2020).
[64] W. Zhang, S. Tople, and O. Ohrimenko, "Dataset-Level Attribute Leakage in Collaborative Learning," Dec. 2020. [Online]. Available: https://arxiv.org/pdf/2006.07267.pdf
[65] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A Study on Overfitting in Deep Reinforcement Learning," 2018. [Online]. Available: https://arxiv.org/pdf/1804.06893
[66] O. Thakkar, S. Ramaswamy, R. Mathews, and F. Beaufays, "Understanding Unintended Memorization in Federated Learning," Jun. 2020. Accessed: Oct. 27 2020. [Online]. Available: https://arxiv.org/pdf/2006.07490.pdf
[67] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can You Really Backdoor Federated Learning?," Nov. 2019. Accessed: Oct. 27 2020. [Online]. Available: https://arxiv.org/pdf/1911.07963.pdf
[68] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning," 12/15/2017. [Online]. Available: https://arxiv.org/pdf/1712.05526.pdf
[69] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How To Backdoor Federated Learning," Feb. 2018. [Online]. Available: https://arxiv.org/pdf/1807.00459.pdf
[70] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified Defenses for Data Poisoning Attacks," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3517–3529, 6755.
[71] *Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks* (accessed: Dec. 30 2020).
[72] *Secret Network.* [Online]. Available: https://scrt.network/about/about-secret-network (accessed: Jan. 8 2021).
[73] Secret Foundation, *Enigma.* [Online]. Available: https://www.enigma.co/ (accessed: Jan. 8 2021).
[74] Secret Foundation, *Network Wiki: How Secret Works* (accessed: Jan. 8 2021).
[75] Ocean Protocol, "Ocean and Secret: Collaborating on Access Control and Private Compute for datatokens," *Ocean Protocol*, 19 Nov., 2020 (accessed: Jan. 8 2021).

[76] M. Bez, G. Fornari, and T. Vardanega, “The scalability challenge of ethereum: An initial quantitative analysis,” in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, San Francisco East Bay, CA, USA, Apr. 2019 - Sep. 2019, pp. 167–176.
[77] *Ethereum blockchain size chart — Blockchair*. [Online]. Available: https://blockchair.com/ethereum/charts/blockchain-size (accessed: Jan. 6 2021).
[78] *State of the DApps — DApp Statistics*. [Online]. Available: https://www.stateofthedapps.com/de/stats (accessed: Jan. 6 2021).
[79] T. Mcconaghy, “Ocean on PoA vs. Ethereum Mainnet? - Ocean Protocol,” *Ocean Protocol*, 12 Feb., 2019 (accessed: Jan. 6 2021).

11. List of Figures

Figure 1: Data Republic Architecture .... - 7 -
Figure 2: Ocean Protocol Architecture .... - 15 -
Figure 3: Ocean Protocol data token exchange overview .... - 25 -
Figure 4: Compute-to-data .... - 27 -
Figure 5: Starting compute-to-data remote compute jobs .... - 29 -
Figure 6: Consume a data asset .... - 34 -
Figure 7: Prototype architecture .... - 35 -
Figure 8: Ocean Market web interface .... - 38 -
Figure 9: VM1 - Infrastructure .... - 38 -
Figure 10: VM2 - Data Provider .... - 39 -
Figure 11: VM3 - Data Consumer .... - 39 -
Figure 12: Web Interface - Publish .... - 40 -
Figure 13: Web Interface - Consume .... - 40 -

12. Appendix

Instruction to set up a full Ocean Protocol stack with default configuration on Ubuntu.

12.1 Prerequisites

```
sudo apt-get install git
sudo apt-get install python3-dev
sudo apt-get install python3-venv (or) python3-virtualenv
```

12.2 Ocean provider

```
git clone git@github.com:oceanprotocol/provider.git
cd provider/
python3 -m venv venv
source venv/bin/activate
pip install Cython
pip install -r requirements_dev.txt
```

edit `/provider/config.ini` with respective parameters

```
network = wss://<network>.infura.io/ws/v3/<id>
dtfactory.address = <address>
aquarius.url = http://127.0.0.1:5000
ocean_provider.url = http://localhost:8030
provider.address = <address>
provider.key = <key>
```

```
export FLASK_APP=ocean_provider/run.py
export CONFIG_FILE=config.ini
export PROVIDER_ADDRESS= <address>
export PROVIDER_KEY= <key>

flask run --port=8030     //make sure to be in virtual environment
```

12.3 Aquarius metadata storage

```
git clone https://github.com/oceanprotocol/aquarius.git
cd aquarius/
```

1. Install requirements

```
sudo apt update
sudo apt install python3-dev python3.7-dev
sudo apt install openjdk-11-jre-headless
```

2. Set up an elasticsearch database

```
export ES_VERSION=6.6.2
export
ES_DOWNLOAD_URL=https://artifacts.elastic.co/downloads/elasticsearch/elasti
csearch-${ES_VERSION}.tar.gz
wget ${ES_DOWNLOAD_URL}
tar -xzf elasticsearch-${ES_VERSION}.tar.gz
sudo chown -R $USER /aquarius/elasticsearch
```

3. Install aquarius server

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
export FLASK_APP=aquarius/run.py
export CONFIG_FILE=config.ini

export EVENTS_RPC=wss://rinkeby.infura.io/ws/v3/<id>
export LOG_LEVEL=DEBUG
export RUN_EVENTS_MONITOR=’1’

edit /aquarius/venv/artifacts/address.json and insert own addresses
```

4. Run

```
./elasticsearch-${ES_VERSION}/bin/elasticsearch & (don’t run as root!)
flask run
```

12.4 Ocean.js

```
git clone https://github.com/oceanprotocol/ocean.js
cd ocean.js
npm install -g install-peerdeps
install-peerdeps eslint-config-airbnb --dev
npm install lib

edit ocean.js/src/utils/ConfigHelper.ts
npm start
```

12.5 React

```
git clone https://github.com/oceanprotocol/react.git
cd react
npm install
npm start
```

12.6 Compute cluster

1. **Activate VT-x or AMD-v.**

   Test if `‘egrep --color 'vmx|svm' /proc/cpuinfo’` is not empty, else retry

2. **Install Docker, Kvm or Virtualbox**

3. **Install Kubernetes cluster: Minikube[15]**

```
curl -Lo minikube
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-
amd64 \ && chmod +x minikube

sudo cp minikube /usr/local/bin && rm minikube
```

4. **Install kubect1[16] command-line tools and connect it to the cluster**

```
curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl
-s https://storage.googleapis.com/kubernetes-
release/release/stable.txt)/bin/linux/amd64/kubectl"

chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
kubectl version --client

minikube start
kubectl cluster-info
```

[15] https://kubernetes.io/de/docs/tasks/tools/install-minikube/
[16] https://kubernetes.io/docs/tasks/tools/install-kubectl/

5. Configure compute-to-data[17]

`mkdir /ocean/operator-service`
Copy config files to `/ocean/operator-service`

```
postgres-configmap.yaml
postgres-storage.yaml
postgres-deployment.yaml
postgres-service.yaml
deployment.yaml
role_binding.yaml
service_account.yaml
```

`mkdir /ocean/operator-engine`
Copy config files to `/ocean/operator-engine`

```
sa.yaml
binding.yaml
operator.yaml
computejob-crd.yaml
workflow-crd.yaml
```

Create namespaces

```
kubectl create ns ocean-operator
kubectl create ns ocean-compute
```

Deploy Operator Service

```
kubectl config set-context --current --namespace ocean-operator
kubectl create -f /ocean/operator-service/postgres-configmap.yaml
kubectl create -f /ocean/operator-service/postgres-storage.yaml
kubectl create -f /ocean/operator-service/postgres-deployment.yaml
kubectl create -f /ocean/operator-service/postgres-service.yaml
kubectl apply -f /ocean/operator-service/deployment.yaml
kubectl apply -f /ocean/operator-service/role_binding.yaml
kubectl apply -f /ocean/operator-service/service_account.yaml
```

Deploy Operator

```
kubectl config set-context --current --namespace ocean-compute
kubectl apply -f /ocean/operator-engine/sa.yml
kubectl apply -f /ocean/operator-engine/binding.yml
kubectl apply -f /ocean/operator-engine/operator.yml
kubectl apply -f /ocean/operator-engine/computejob-crd.yaml
kubectl apply -f /ocean/operator-engine/workflow-crd.yaml
kubectl create -f /ocean/operator-service/postgres-configmap.yaml
```

[17] https://docs.oceanprotocol.com/tutorials/compute-to-data/

Expose Operator Service

```
kubectl expose deployment operator-api --namespace=ocean-operator --
port=8050

kubectl -n ocean-operator port-forward svc/operator-api 8050

curl -X POST "http://example.com:8050/api/v1/operator/pgsqlinit" -H
"accept: application/json"
```

12.7 Ocean market

```
git clone https://github.com/oceanprotocol/market.git
cd market
npm install -g npm
sudo chown -R $USER /home/ocean/market/
// switch to non-root user
npm install
npm start
```

12.8 Ocean smart contracts

1. Register at Infura[18] to get an Infura ID

2. Deploy contracts

```
git clone https://github.com/oceanprotocol/contracts.git
cd contracts
sudo chown -R $USER /home/ocean/contracts/node_modules…
npm i

sudo export MNEMONIC='YOUR MNEMONIC HERE'
sudo export INFURA_TOKEN='YOUR INFURA ID HERE’
```

edit `/contracts/truffle.js` and insert the first address of the mnemonic of the network to deploy to

make sure the address has sufficient funds!

```
sudo npm run deploy:<network>
```

[18] www.infura.io

12.8 Barge

To test on a local environment first, one can use Barge. It incorporates the full Ocean Protocol stack with a local Ethereum test-net based on Truffle Suite[19].

1. **Install Docker[20]**

```
sudo apt-get install \ apt-transport-https \ ca-certificates \ curl \
    gnupg-agent \software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
–
sudo apt-key fingerprint 0EBFCD88

pub   rsa4096 2017-02-22 [SCEA]
      9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid           [ unknown] Docker Release (CE deb) <docker@docker.com>
sub   rsa4096 2017-02-22 [S]          // check if fingerprints match


sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \stable"

sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
sudo docker run hello-world
```

2. **Install Docker Compose[21]**

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.27.4/docker-compose-
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose
$ docker-compose –version
```

3. **Install Barge**

```
git clone https://github.com/oceanprotocol/barge.git
cd barge

cp .env.example .env
./start_ocean.sh
```

[19] https://www.trufflesuite.com/
[20] https://docs.docker.com/engine/install/ubuntu/
[21] https://docs.docker.com/compose/install/