

Composable DeSci Flows Via Token-Wrapped Data & Algorithms

Trent McConaghy



ocean

oceanprotocol.com

[@trentmc0](https://twitter.com/trentmc0)

Science * Data

Science is about making models to predict unseen events, in a reproducible way.

It's data all the way down.

Data can be

- raw measurements
- cleaned data
- features
- scientific models
- predictions

Data can also be algorithms to build the models.

Data can be dynamically changing.



Tokens for Composable Data Flows

Goal: composable, reproducible data flows.

How: tokenize data & algorithms

- **Small data: store on-chain inside ERC721 data NFT**
- **Large data: store off-chain. Access control via ERC20 datatokens**
 - Compute-to-Data flows preserve privacy and control.
 - Data & algorithms can use Web3 storage or compute.

Details: Where to store data <> How to share it

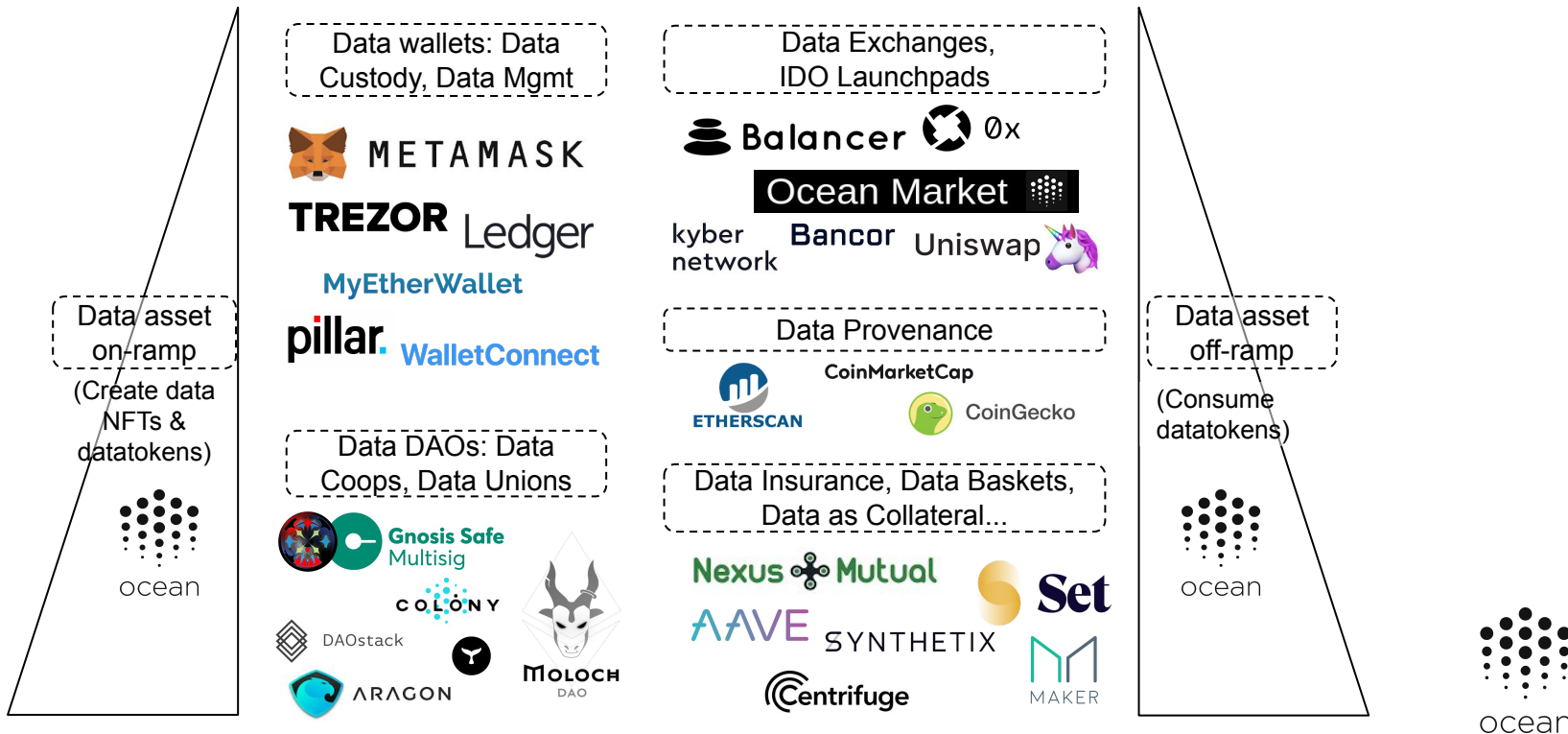
Where to store	Where to store: specific medium	How to share (access control)
Off-chain	Any web2 or web3 service. Eg S3, Filecoin	<ul style="list-style-type: none">● Fully open, don't need provenance: just use http● Fully open, want provenance: Ocean datatokens, with free dispense● Share if paid: Ocean datatokens, with fixed-price, AMM, etc.● Fully private, only seen by algorithms: Ocean datatokens + Compute-to-Data
On-chain (small data)	Key-value pairs in data NFTs	<ul style="list-style-type: none">● Fully open: store value plaintext● Open to marketplaces etc: encrypt value, share symmetric key liberally● Open sparingly: encrypt value, share symmetric key sparingly

Tokenize Data & Algorithms

Data on-ramp: deploy and mint ERC721 data NFTs & ERC20 datatokens.

Data off-ramp: consume datatokens

Enables data assets * Web3 wallets, exchanges, and DAOs



Atomic → Higher Level Building Blocks

Atomic building blocks: Data NFTs and datatokens

Higher level blocks. The atomic blocks naturally interoperate with

- Web3 wallets
- DAOs
- DEXes
- NFT marketplaces
- etc

Higher level yet. From this, we can construct many DeSci flows:

- AI-training provenance
- scientific model commons
- algorithm marketplaces
- and more

What I'll cover in detail

- On-chain data → data NFTs
- On-chain data with privacy → data NFTs with encryption
- Off-chain data → datatokens
- Off-chain data with privacy → datatokens with Compute-to-Data



On-chain data: Data NFTs

On-chain data (small): Ocean Data NFTs

```
erc721_nft = ocean.create_erc721_nft('NFTToken1', 'NFT1', alice_wallet)
```

```
#Key-value pair
key = "fav_color"
value = "blue"

#prep key for setter
key_hash = ocean.web3.keccak(text=key) #Contract/ERC725 requires keccak256 hash

#prep value for setter
value_hex = value.encode('utf-8').hex() #set_new_data() needs hex

#set
erc721_nft.set_new_data(key_hash, value_hex, alice_wallet)
```

```
value2_hex = erc721_nft.get_data(key_hash)
value2 = value2_hex.decode('ascii')
print(f"Found that {key} = {value2}")
```





On-chain data with privacy: Data NFTs with encryption

On-chain data with privacy: Data NFTs with encryption

Basic idea: symmetric key to encrypt & decrypt; share it by encrypting with consumer's pub key

- Use cases: “login with Web3”, “soul-bound tokens”
- <https://github.com/oceanprotocol/ocean.py/blob/v4main/READMEs/profile-nfts-flow.md>

```
erc721_nft = ocean.create_erc721_nft('NFTToken1', 'NFT1', alice_wallet)

#Prep value for setter
profiledata_val_encr_hex = Fernet(symkey).encrypt(profiledata_val.encode('utf-8')).hex()

#set
erc721_nft.set_new_data(profiledata_name_hash, profiledata_val_encr_hex, alice_wallet)

profiledata_val_encr_hex2 = erc721_nft.get_data(profiledata_name_hash)
profiledata_val2_bytes = Fernet(symkey).decrypt(profiledata_val_encr_hex2)
```



Off-chain data: Datatokens

Off-chain data:

Ocean datatokens

```
erc721_nft = ocean.create_erc721_nft('NFTToken1', 'NFT1', alice_wallet)
```

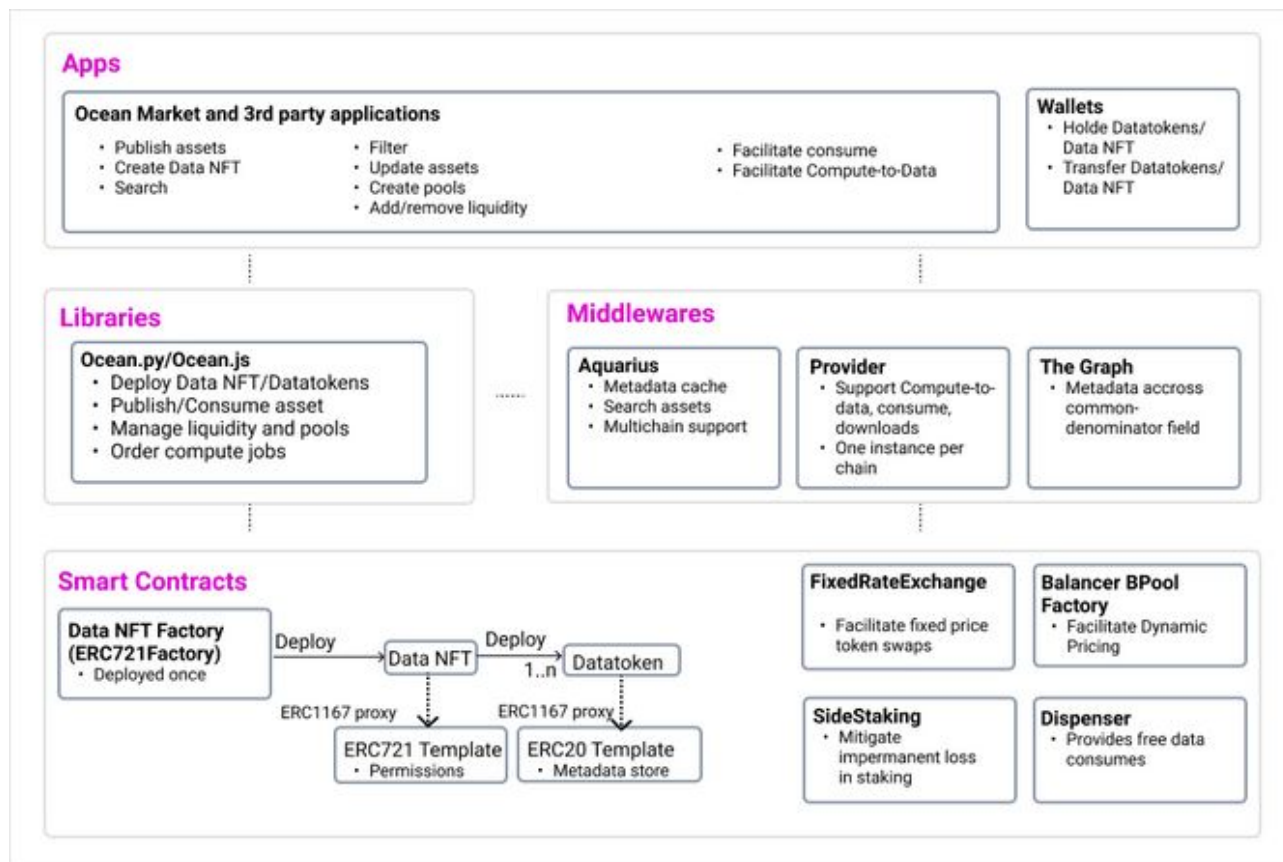
```
erc20_token = erc721_nft.create_datatoken(  
    template_index=1, # default value  
    name="ERC20DT1", # name for ERC20 token  
    symbol="ERC20DT1Symbol", # symbol for ERC20 token  
    minter=alice_wallet.address, # minter address
```

<here: Bob gets 1.0 datatokens>

```
# Bob sends his datatoken to the service  
order_tx_id = ocean.assets.pay_for_access_service(  
    asset,  
    service,
```

```
# Bob downloads. If the connection breaks, Bob can request again by  
file_path = ocean.assets.download_asset(  
    asset=asset,  
    service=service,  
    consumer_wallet=bob_wallet,  
    destination='./',  
    order_tx_id=order_tx_id  
)
```

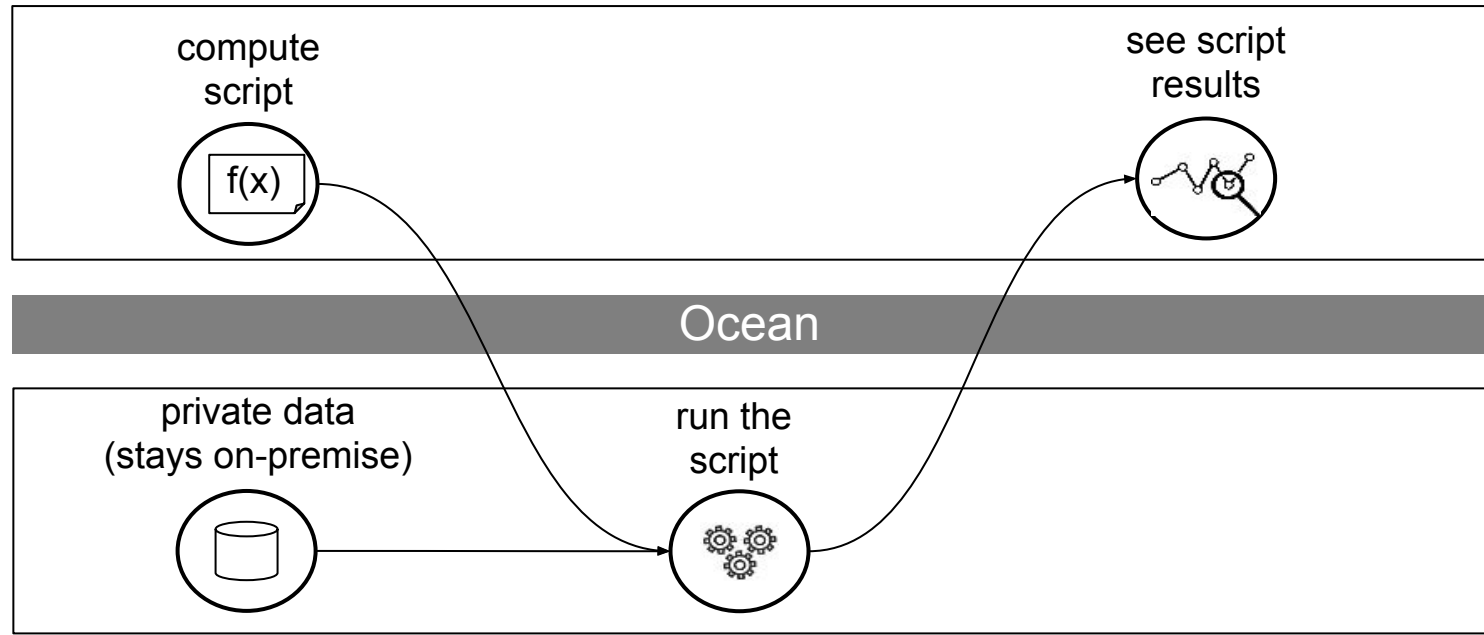
Detail: Ocean architecture





Off-chain data with privacy: Datatokens + Compute-to-Data

Off-chain data with privacy: Ocean datatokens with Compute-to-Data



C2D Quickstart via Ocean.py: Overview

github.com/oceanprotocol/ocean.py/blob/main/READMEs/c2d-flow.md

Quickstart

Simple Flow

This stripped-down flow shows the essence of Ocean: simply create

[Go to simple flow](#)

Marketplace flow

In this flow, a data asset is posted for sale in a marketplace, and pool.

[Go to marketplace flow](#)

Compute-to-Data flow

This flow uses Ocean Compute-to-Data (c2d) to compute results

[Go to c2d flow](#)



3. Alice publishes algorithm

For this step, there are some prerequisites needed. If you want to replace the sample algo need to do some dependency management. You can use one of the standard [Ocean algo](#). Use the image name and tag in the `container` part of the algorithm metadata. This dock dependency installation e.g. in the case of Python, OS-level library installations, pip install more about docker image publishing.

In the same Python console:

```
# Publish ALG datatoken
ALG_datatoken = ocean.create_data_token('ALG1', 'ALG1', alice_wallet, blob=
ALG_datatoken.mint(alice_wallet.address, to_wei(100), alice_wallet)
print(f"ALG_datatoken.address = '{ALG_datatoken.address}'")

# Specify metadata and service attributes, for "GPR" algorithm script.
# In same location as Branim test dataset. GPR = Gaussian Process Regression
ALG_metadata = {
    "main": {
        "type": "algorithm",
```



6. Bob starts a compute job

Only inputs needed: DATA_did, ALG_did. Everything else can get computed as ne

In the same Python console:

```
DATA_did = DATA_ddo.did # for convenience
ALG_did = ALG_ddo.did
DATA_DDO = ocean.assets.resolve(DATA_did) # make sure we operate on t
ALG_DDO = ocean.assets.resolve(ALG_did)

compute_service = DATA_DDO.get_service('compute')
algo_service = ALG_DDO.get_service('access')

from ocean_lib.web3_internal.constants import ZERO_ADDRESS
from ocean_lib.models.compute_input import ComputeInput

# order & pay for dataset
```

You can use the result however you like. For the purpose of this example, let's plot it.

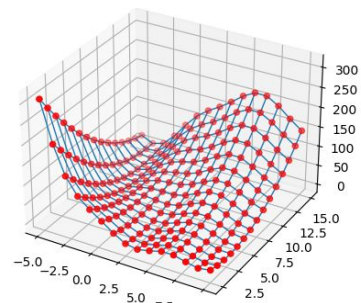
```
import numpy
from matplotlib import pyplot

X0_vec = numpy.linspace(-5., 10., 15)
X1_vec = numpy.linspace(0., 15., 15)
X0, X1 = numpy.meshgrid(X0_vec, X1_vec)
b, c, t = 0.12918450914398066, 1.5915494309189535, 0.039788735772973836
u = X1 - b*X0**2 + c*X0 - 6
r = 10.*(1 - t) * numpy.cos(X0) + 10
Z = u**2 + r

fig, ax = pyplot.subplots(subplot_kw={"projection": "3d"})
ax.scatter(X0, X1, model, c="r", label="model")
pyplot.title("Data + model")
pyplot.show() # or pyplot.savefig("test.png") to save the plot as a .png file
```



Data + model



Here are the steps:

1. Setup
2. Alice publishes data asset
3. Alice publishes algorithm
4. Alice allows the algorithm for C2D for that data asset
5. Bob acquires datatokens for data and algorithm
6. Bob starts a compute job
7. Bob monitors logs / algorithm output

C2D Quickstart via Ocean.py: Where to find

github.com/oceanprotocol/ocean.py/blob/main/READMEs/c2d-flow.md

Quickstart

Here are flows to try out, from simple to specific detailed variants.

- **Simple flow** - the essence of Ocean - creating a data NFT & datatoken.
- **Publish flow** - a dataset is published.
- **Consume flow** - a published dataset is consumed (downloaded).
- **Marketplace flow** - a data asset is posted for sale in a datatoken pool, then purchased. Includes metadata.
- **Fixed rate exchange flow** - a data asset is posted for sale at fixed rate, then purchased.
- **Dispenser flow** - here, a datatoken dispenser is created and datatokens are dispensed for free.
- **Compute-to-data flow** - uses C2D to build an AI model a dataset that never leaves the premises.
- **Key-value database** - use data NFTs to store arbitrary key-value pairs on-chain.
- **Profile NFTs** - enable "login with Web3" where Dapp can access private user profile data.

C2D Quickstart: Steps

github.com/oceanprotocol/ocean.py/blob/main/READMEs/c2d-flow.md

1. Setup
2. Alice publishes data asset
3. Alice publishes algorithm
4. Alice allows the algorithm for C2D for that data asset
5. Bob acquires datatokens for data and algorithm
6. Bob starts a compute job
7. Bob monitors logs / algorithm output

C2D Quickstart: Step 2: Publish dataset

github.com/oceanprotocol/ocean.py/blob/main/READMEs/c2d-flow.md

```
# Publish DATA datatoken, mint tokens
from ocean_lib.web3_internal.currency import to_wei

DATA_datatoken = ocean.create_data_token('DATA1', 'DATA1', alice_wallet, blob=ocean.config.metadata_cache_uri)
DATA_datatoken.mint(alice_wallet.address, to_wei(100), alice_wallet)
print(f"DATA_datatoken.address = '{DATA_datatoken.address}'")

# Specify metadata & service attributes for Branin test dataset.
# It's specified using _local_ DDO metadata format; Aquarius will convert it to remote
# by removing `url` and adding `encryptedFiles` field.
DATA_metadata = {
    "main": {
        "type": "dataset",
        "files": [
            {
                "url": "https://raw.githubusercontent.com/trentmc/branin/main/branin.arff",
                "index": 0,
                "contentType": "text/text"
            }
        ]
    }
},
```

C2D Quickstart: Step 3: Publish algorithm

github.com/oceanprotocol/ocean.py/blob/main/READMEs/c2d-flow.md

```
# Publish ALG datatoken
ALG_datatoken = ocean.create_data_token('ALG1', 'ALG1', alice_wallet, blob=ocean.config.metadata_cache_uri)
ALG_datatoken.mint(alice_wallet.address, to_wei(100), alice_wallet)
print(f"ALG_datatoken.address = '{ALG_datatoken.address}'")

# Specify metadata and service attributes, for "GPR" algorithm script.
# In same location as Branin test dataset. GPR = Gaussian Process Regression.
ALG_metadata = {
    "main": {
        "type": "algorithm",
        "algorithm": {
            "language": "python",
            "format": "docker-image",
            "version": "0.1",
            "container": {
                "entrypoint": "python $ALGO",
                "image": "oceanprotocol/algo_dockers",
                "tag": "python-branin"
            }
        }
    },
    "files": [
        {
            "url": "https://raw.githubusercontent.com/trentmc/branin/main/gpr.py",
            "index": 0.
```

C2D Quickstart: Step 4: dataset allows algorithm

github.com/oceanprotocol/ocean.py/blob/main/READMEs/c2d-flow.md

```
from ocean_lib.assets.trusted_algorithms import add_publisher_trusted_algorithm
add_publisher_trusted_algorithm(DATA_ddo, ALG_ddo.did, config.metadata_cache_uri)
ocean.assets.update(DATA_ddo, publisher_wallet=alice_wallet)
```

C2D Quickstart: Step 5: get data & alg assets

github.com/oceanprotocol/ocean.py/blob/main/READMEs/c2d-flow.md

```
bob_wallet = Wallet(  
    ocean.web3,  
    os.getenv('TEST_PRIVATE_KEY2'),  
    config.block_confirmations,  
    config.transaction_timeout,  
)  
print(f"bob_wallet.address = '{bob_wallet.address}')
```


Alice shares access for both to Bob, as datatokens. Alternatively, Bob might have bought these in a market.

```
DATA_datatoken.transfer(bob_wallet.address, to_wei(5), from_wallet=alice_wallet)  
ALG_datatoken.transfer(bob_wallet.address, to_wei(5), from_wallet=alice_wallet)
```


C2D Quickstart: Step 6: start compute

github.com/oceanprotocol/ocean.py/blob/main/READMEs/c2d-flow.md

```
# order & pay for dataset
dataset_order_requirements = ocean.assets.order(
    DATA.did, bob_wallet.address, service_type=compute_service.type
)
DATA_order_tx_id = ocean.assets.pay_for_service(
    ocean.web3,
    dataset_order_requirements.amount,
    dataset_order_requirements.data_token_address,
    DATA.did,
    compute_service.index,
    ZERO_ADDRESS,
    bob_wallet,
    dataset_order_requirements.computeAddress,
)
```



```
# order & pay for algo
algo_order_requirements = ocean.assets.order(
    ALG.did, bob_wallet.address, service_type=algo_service.type
)
ALG_order_tx_id = ocean.assets.pay_for_service(
    ocean.web3,
    algo_order_requirements.amount,
    algo_order_requirements.data_token_address,
    ALG.did,
    algo_service.index,
    ZERO_ADDRESS,
    bob_wallet,
    algo_order_requirements.computeAddress,
)
```



```
compute_inputs = [ComputeInput(DATA.did, DATA_order_tx_id, compute_service.index)]
job_id = ocean.compute.start(
    compute_inputs,
    bob_wallet,
    algorithm.did=ALG.did,
    algorithm.tx_id=ALG_order_tx_id,
    algorithm_data_token=ALG_data_token.address
)
print(f"Started compute job with id: {job_id}")
```


C2D Quickstart: step 7: see output

github.com/oceanprotocol/ocean.py/blob/main/READMEs/c2d-flow.md

In the same Python console, you can check the job status as many times as needed:

```
ocean.compute.status(DATA_did, job_id, bob_wallet)
```

This will output the status of the current job. Here is a list of possible results: [Operator Service Status description](#).

Once you get `{'ok': True, 'status': 70, 'statusText': 'Job finished'}`, Bob can check the result of the job.

```
result = ocean.compute.result_file(DATA_did, job_id, 0, bob_wallet) # 0 index, means we retrieve the
import pickle
model = pickle.loads(result) # the gaussian model result
```

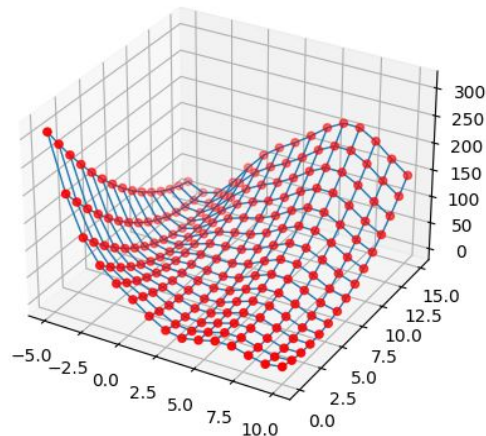
You can use the result however you like. For the purpose of this example, let's plot it.

```
import numpy
from matplotlib import pyplot

X0_vec = numpy.linspace(-5., 10., 15)
X1_vec = numpy.linspace(0., 15., 15)
X0, X1 = numpy.meshgrid(X0_vec, X1_vec)
b, c, t = 0.12918450914398066, 1.5915494309189535, 0.039788735772973836
u = X1 - b*X0**2 + c*X0 - 6
r = 10.*(1. - t) * numpy.cos(X0) + 10
Z = u**2 + r

fig, ax = pyplot.subplots(subplot_kw={"projection": "3d"})
ax.scatter(X0, X1, model, c="r", label="model")
pyplot.title("Data + model")
pyplot.show() # or pyplot.savefig("test.png") to save the plot as a .png file
```

Data + model



C2D Quickstart via Ocean.py: Recap

github.com/oceanprotocol/ocean.py/blob/main/READMEs/c2d-flow.md

Quickstart

Simple Flow

This stripped-down flow shows the essence of Ocean: simply create a data asset and an algorithm.

[Go to simple flow](#)

Marketplace flow

In this flow, a data asset is posted for sale in a marketplace, and a pool.

[Go to marketplace flow](#)

Compute-to-Data flow

This flow uses Ocean Compute-to-Data (c2d) to compute results from a data asset and an algorithm.

[Go to c2d flow](#)

3. Alice publishes algorithm

For this step, there are some prerequisites needed. If you want to replace the sample algorithm, you need to do some dependency management. You can use one of the standard [Ocean algorithms](#). Use the image name and tag in the `container` part of the algorithm metadata. This docker dependency installation e.g. in the case of Python, OS-level library installations, pip install more about docker image publishing.

In the same Python console:

```
# Publish ALG datatoken
ALG_datatoken = ocean.create_data_token('ALG1', 'ALG1', alice_wallet, blob=ALG_blob)
ALG_datatoken.mint(alice_wallet.address, to_wei(100), alice_wallet)
print(f"ALG_datatoken.address = '{ALG_datatoken.address}'")

# Specify metadata and service attributes, for "GPR" algorithm script.
# In same location as Branin test dataset. GPR = Gaussian Process Regression
ALG_metadata = {
    "main": {
        "type": "algorithm",
```

You can use the result however you like. For the purpose of this example, let's plot it.

```
import numpy
from matplotlib import pyplot

X0_vec = numpy.linspace(-5., 10., 15)
X1_vec = numpy.linspace(0., 15., 15)
X0, X1 = numpy.meshgrid(X0_vec, X1_vec)
b, c, t = 0.12918450914398066, 1.5915494309189535, 0.039788735772973836
u = X1 - b*X0**2 + c*X0 - 6
r = 10.*(1 - t) * numpy.cos(X0) + 10
Z = u**2 + r

fig, ax = pyplot.subplots(subplot_kw={"projection": "3d"})
ax.scatter(X0, X1, model, c="r", label="model")
pyplot.title("Data + model")
pyplot.show() # or pyplot.savefig("test.png") to save the plot as a .png file
```

Here are the steps:

1. Setup
2. Alice publishes data asset
3. Alice publishes algorithm
4. Alice allows the algorithm for C2D for that data asset
5. Bob acquires datatokens for data and algorithm
6. Bob starts a compute job
7. Bob monitors logs / algorithm output

6. Bob starts a compute job

Only inputs needed: DATA_did, ALG_did. Everything else can get computed as needed.

In the same Python console:

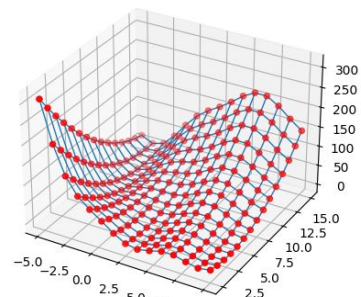
```
DATA_did = DATA_ddo.did # for convenience
ALG_did = ALG_ddo.did
DATA_DDO = ocean.assets.resolve(DATA_did) # make sure we operate on the correct DDO
ALG_DDO = ocean.assets.resolve(ALG_did)

compute_service = DATA_DDO.get_service('compute')
algo_service = ALG_DDO.get_service('access')

from ocean_lib.web3_internal.constants import ZERO_ADDRESS
from ocean_lib.models.compute_input import ComputeInput

# order & pay for dataset
```

Data + model



Compute-to-Data In Ocean Market

blog.oceanprotocol.com/compute-to-data-is-now-available-in-ocean-market-58868be52ef7

market.oceanprotocol.com/publish

Ocean Market PUBLISH PROFILE

Publish

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

Given the beta status, publishing on Ropsten or Rinkeby first is strongly recommended. Please familiarize yourself with the [market](#), the [risks](#), and the [Terms of Use](#).

Publish on Algorithm

Title*

e.g. Shapes of Desert Plants

Enter a concise title.

Random Forest Classifier v1.0

Polygon

ALGORITHM | Friable Nautilus Token – FRINAU-65

Published By [OxAcca..1f83](#)
28 days ago

Random forest is a supervised algorithm. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

random-forest classifier

DATA AUTHOR: Raven Protocol & Ocean Protocol

OWNER: [OxAcca..1f83](#)

DOCKER IMAGE: oceanprotocol/algo_dockers:python-panda

DID: d1d:op:e64810f2A54359198026cEaC2F38545F65D672f0

Ocean Market PUBLISH PROFILE

31 results

DATA SETS ALGORITHMS DOWNLOAD COMPUTE Clear

Sort Relevance Published

- AEAPND Apply pandas filter
- SPASHA-36 Logistic Regression v1.0
- FRINAU-65 Random Forest Regressor v1.0

Job finished

Daily Fishing Effort (01.01.2020)

ARTRAV-68 | did:op:2f633a67aD3e6d59c5235546752c4082F10672EF

Count data points (fixed price)

EFFTUR-46 | did:op:2E761103C6A52F13998B438F6c65d56303F884698

GET RESULTS

Results are stored for 30 days.

CREATED: 8 days ago FINISHED: 8 days ago

JOB ID: 342b24b3c41b4afdb0c24493f4aa188c



ocean



Ocean Market: Decentralized data market for algorithms + data

Ocean Market: Splash Page

market.oceanprotocol.com

We are in beta. Please familiarize yourself with [the market](#), [the risks](#), and the [Terms of Use](#).

Ocean Market **v3** PUBLISH PROFILE

Search...

Ocean Market

A marketplace to find, publish and trade data sets in the Ocean Network.

Bookmarks


Your bookmarks will appear here.

Highest Liquidity

QUICRA-0	LUMSTA-42	EXCANE-93
DataUnion.app - Image & Annotation Vault DataUnion.app Notice This dataset and the software stack behind it are under constant de...	Product Pages of 1'044'709 Products on Amazon.com (pro... Innovation Atelier SA Result of scraping of Amazon.com product page data over H1 2018, obtai...	EVO/2MP/TRFC/DE/200K Weekly Collector Evotegra GmbH Evotegra - EVO/2MP/TRFC/DE/200K German Traffic Data for Machine Lear...
210.354 OCEAN POOL	82,862.46 OCEAN POOL	1,511.099 OCEAN POOL



Ocean Market: Publish Flow, for an Initial Data Offering



PUBLISH

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*

Enter a concise title.

Description*

Add a thorough description with as much detail as possible. You can use [Markdown](#).

File*

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

Sample file

Please provide a URL to a sample of your data set file. This file should reveal the data structure of your data set, e.g. by including the header and one line of a CSV file. This file URL will be publicly available after publishing.

Access Type*

Please provide a URL to a sample of your data set file. This file should reveal the data structure of your data set, e.g. by including the header and one line of a CSV file. This file URL will be publicly available after publishing.

Access Type*

Choose how you want your files to be accessible for the specified price.

Datatoke Name & Symbol*

The datatoken for this data set will be created with this name & symbol.

Author*

Give proper attribution for your data set.

Tags

Separate tags with comma.

Terms & Conditions*

Ocean Marketplace Terms and Conditions (this "Agreement") is made and entered into by and between Ocean Protocol Foundation Ltd., with office at The Commerz @ Irving, 1 Irving Place, #08-11, Singapore, 369546 Singapore ("Ocean") and the legal entity set forth in the Account Information ("Customer"). It governs Customer's access to and use of the Ocean Marketplace (as defined below) and takes effect on the date of its acceptance by Customer (the "Effective Date"). Customer represents being lawfully able to enter into contracts and having legal authority to bind Customer's entity.

DEFINITIONS

"Service" means all websites, software and services offered and operated by

I agree to these Terms and Conditions

Example Data Asset, for Fixed Price



Ocean Market **BETA**

PUBLISH

HISTORY

Get MetaMask



eBay DATASET - 10 Million Data Points (1,000,000 Product Listings)

Data Reservoir

Exceptional Whale Token – EXCWHA-70 ↗
Published by [0x98EA...16E4](#) – Etherscan ↗



This dataset has a massive total of over 10 million data points from over 1,000,000 product listings on eBay using the electronics category. This dataset is from the first week of November 2020.

- Updated monthly

What's included in the dataset?

The dataset is in xlsx format and each line shows 10 data points with the date & time scraped. The following is included in this dataset:

- *Seller name
- *Seller rating
- *Item category
- *Item ID

USE

POOL

TRADE



2,639.166 OCEAN **POOL**
= €1,211.94

BUY

For using this data set, you will buy 1 EXCWHA-70 and immediately spend it back to the publisher and pool.

■ **No account connected**

Please connect your Web3 wallet.



Example Data Asset, with Automatic Price Discovery (via AMM)

Ocean Market BETA

PUBLISH HISTORY [Get MetaMask](#)

AtlantisStream.io - Realtime Consumer Data Streams

Atlantis Streams

Meretricious Manatee Token – MERMAN-13
Published by 0x4f40_50B3 – Etherscan

Atlantis Stream is a crowdsourced dataset of real-time consumer data streams.

Notice (11/17/2020)

Atlantis Stream is currently pre-alpha, and will be migrating to compute-to-data when it becomes available. Stay up to date on any of our official channels below:

- Website
- Newsletter
- Telegram
- Twitter
- Discord
- Github

For business inquires:

- Contact [our founder](#)
- Email us at team@atlantisstream.io

How it works.

USE POOL TRADE

No file info available

289.698 OCEAN POOL
= €132.88

BUY

For using this data set, you will buy 1 MERMAN-13 and immediately spend it back to the publisher and pool.

No account connected
Please connect your Web3 wallet.

Example Data Asset: A Data Union

The screenshot displays the Ocean Market interface. At the top left is the Ocean Market logo with a 'BETA' badge. Navigation links for 'PUBLISH' and 'HISTORY' are visible, along with a 'Connect Wallet' button and a settings icon. The main heading is 'Swash - Consumer Browsing Data'. The asset listing includes the publisher 'SwashData Tech Oy', the token 'Tasty Lobster Token - TASLOB-45', and publication details. A description explains that Swash crowdsources user surfing data and shares profits. 'Use cases' include market intelligence and advertising optimization. An update from November 2020 lists 800K data points and 1600 members. A right-hand panel shows the asset's price as 31,958.954 OCEAN (€14,448.80) and a 'BUY' button. A 'No account connected' message prompts the user to connect their Web3 wallet.

Ocean Market BETA

PUBLISH HISTORY [Connect Wallet](#) ⚙️

Swash - Consumer Browsing Data

SwashData Tech Oy

Tasty Lobster Token – TASLOB-45 ↗
Published by SwashData Tech Oy – [Home](#) ↗ [Twitter](#) ↗ [Etherscan](#) ↗

Swash is creating the world's first **Data Union**. It crowdsources users' surfing data through a browser plugin (available on Chrome, Firefox, Brave, Edge, and more) and shares profits with users. This lets Swash provide data buyers with unrivaled zero-party consumer data at scale, from all over the web, guaranteeing data quality and user consent. The increasing number of users will grow the value of Swash data assets over time.

Use cases

Market intelligence, Consumer insights, E-commerce analytics, AI/ML, and Advertising optimisation

UPDATE: November 21th 2020:

- Number of data points: 800K (+100k since last update)
- Data Union members: 1600 (+100 since last update)
- Coverage: Worldwide

USE POOL TRADE

zip
68.85 MB

31,958.954 OCEAN POOL
= €14,448.80

BUY

For using this data set, you will buy 1 TASLOB-45 and immediately spend it back to the publisher and pool.

No account connected
Please connect your Web3 wallet.

Ocean is multi-chain

blog.oceanprotocol.com/ocean-makes-multinetwork-even-simpler-c3ec6c0cbd50

The screenshot shows the Ocean Market website interface. At the top, there is a navigation bar with the Ocean Market logo, 'PUBLISH', and 'PROFILE' buttons. A search bar and a user profile dropdown (Ox7F5dc...5497) are also present. A pink arrow points to a 'Networks' dropdown menu that is open, showing a list of blockchain networks. The 'Main' section includes ETH, Polygon, and BSC, all of which are checked. The 'Test' section includes ETH Ropsten, ETH Rinkeby, Polygon Mumbai, Moonbase Alpha, and GAIA-X Testnet, all of which are unchecked. Below the network selection, the main content area features the 'Ocean Market' title and a subtitle: 'A marketplace to find, publish and trade data sets in the Ocean Net'. There are also sections for 'Bookmarks' and 'Highest Liquidity'. At the bottom, three data set cards are displayed, each with a title, description, and price in OCEAN POOL.

market.oceanprotocol.com

We are in beta. Please familiarize yourself with [the market](#), [the risks](#), and the [Terms of Use](#).

Ocean Market **v3** PUBLISH PROFILE

Search...

Ox7F5dc...5497

Ocean Market

A marketplace to find, publish and trade data sets in the Ocean Net

Bookmarks
Your bookmarks will appear here.

Highest Liquidity

Networks
Switch the data source for the interface.

Main

- ETH
- Polygon
- BSC

Test

- ETH Ropsten
- ETH Rinkeby
- Polygon Mumbai
- Moonbase Alpha
- GAIA-X Testnet

↓ DATA SET

QUICRA-0
DataUnion.app - Image & Annotation Vault
DataUnion.app
Notice This dataset and the software stack behind it are under constant development...
692.868 OCEAN POOL

↓ DATA SET

LUMSTA-42
Product Pages of 1'044'709 Products on Amazon.com (proces...
Innovation Atelier SA
Result of scraping of Amazon.com product page data over H1 2018, obtained using ne...
83,942.031 OCEAN POOL

↓ DATA SET

EXCANE-93
EVO/2MP/TRFC/DE/200K Weekly Collector
Evotegra GmbH
Evotegra - EVO/2MP/TRFC/DE/200K German Traffic Data for Machine Learning ...
1,523.022 OCEAN POOL

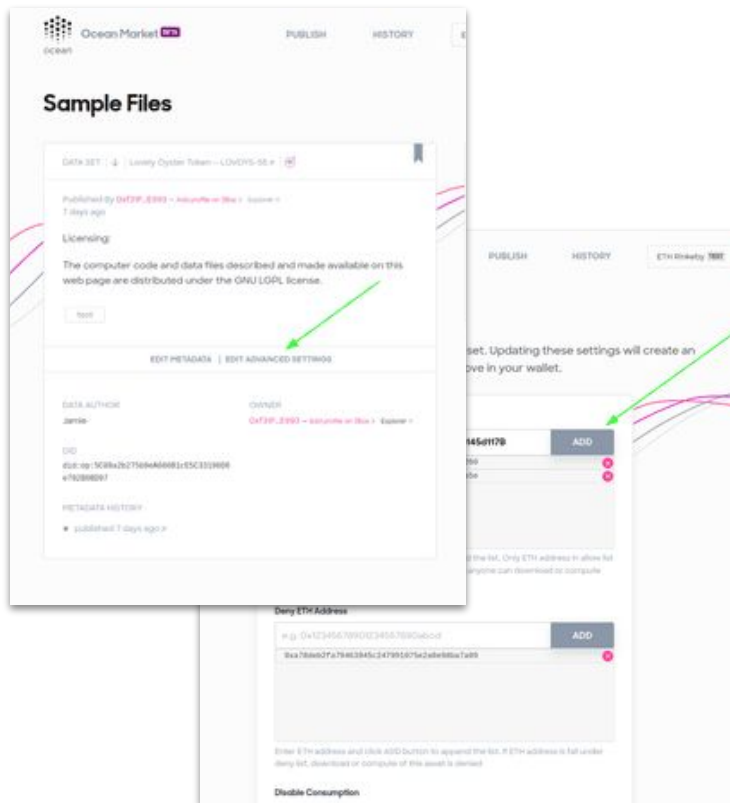
Fine-grained permissions

blog.oceanprotocol.com/fine-grained-permissions-now-supported-in-ocean-protocol-4fe434af24b9

How to handle data exchange for

- 🏥 Medical data only for credentialed EU researchers
- 🚗 Selling automotive data within a consortium
- 🇩🇪🇮🇹 Sharing data across offices in a multinational ?

@oceanprotocol fine-grained permissions handles this





Conclusion

Science * Data

Science is about making models to predict unseen events, in a reproducible way.

It's data all the way down.

Data can be

- raw measurements
- cleaned data
- features
- scientific models
- predictions

Data can also be algorithms to build the models.

Data can be dynamically changing.



Tokens for Composable Data Flows

Goal: composable, reproducible data flows.

How: tokenize data & algorithms

- **Small data: store on-chain inside ERC721 data NFT**
- **Large data: store off-chain. Access control via ERC20 datatokens**
 - Compute-to-Data flows preserve privacy and control.
 - Data & algorithms can use Web3 storage or compute.

Tokens for Composable Data Flows

Try for yourself!

- <https://github.com/oceanprotocol/ocean.py>
- <https://docs.oceanprotocol.com>