

Composable DeSci Flows Via Token-Wrapped Data & Algorithms

Trent McConaghy



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.



ocean

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

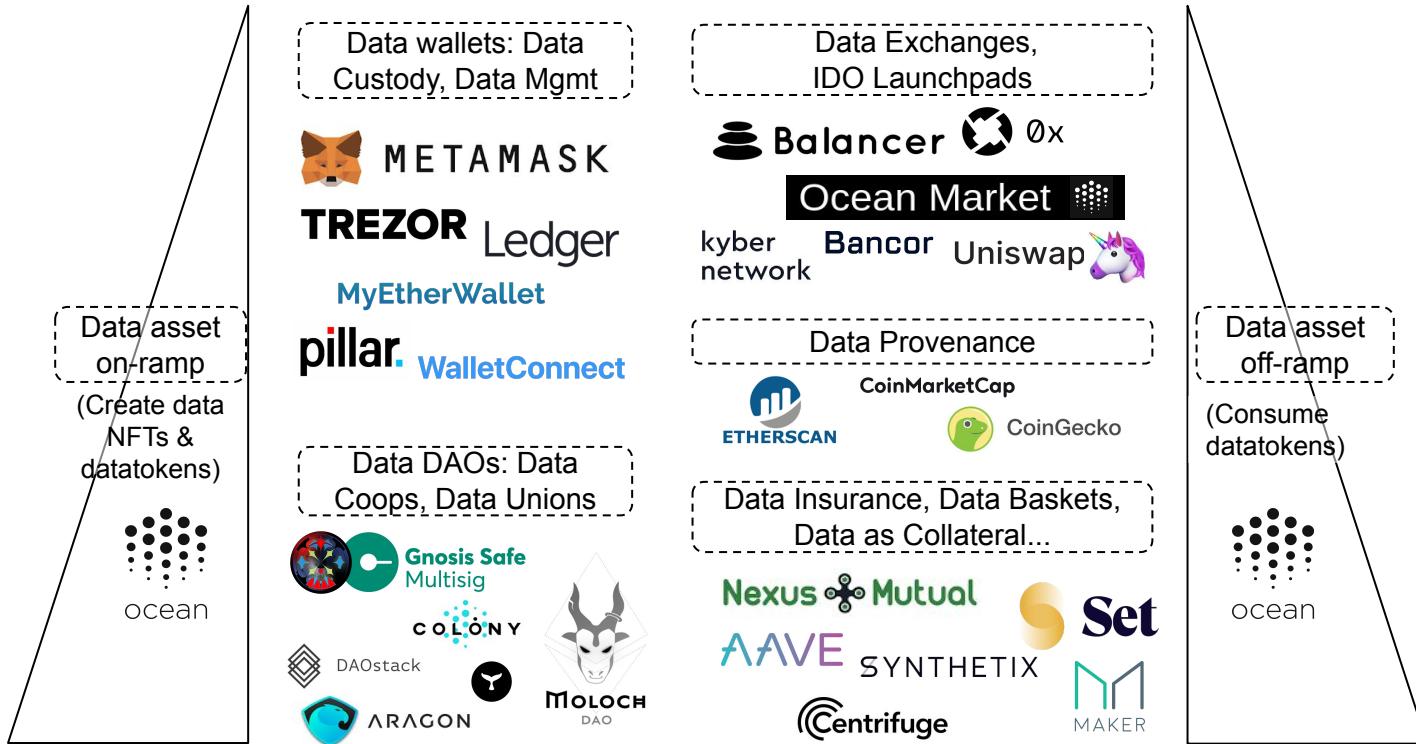


Tokenize Data & Algorithms

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}")
```



ocean



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
```

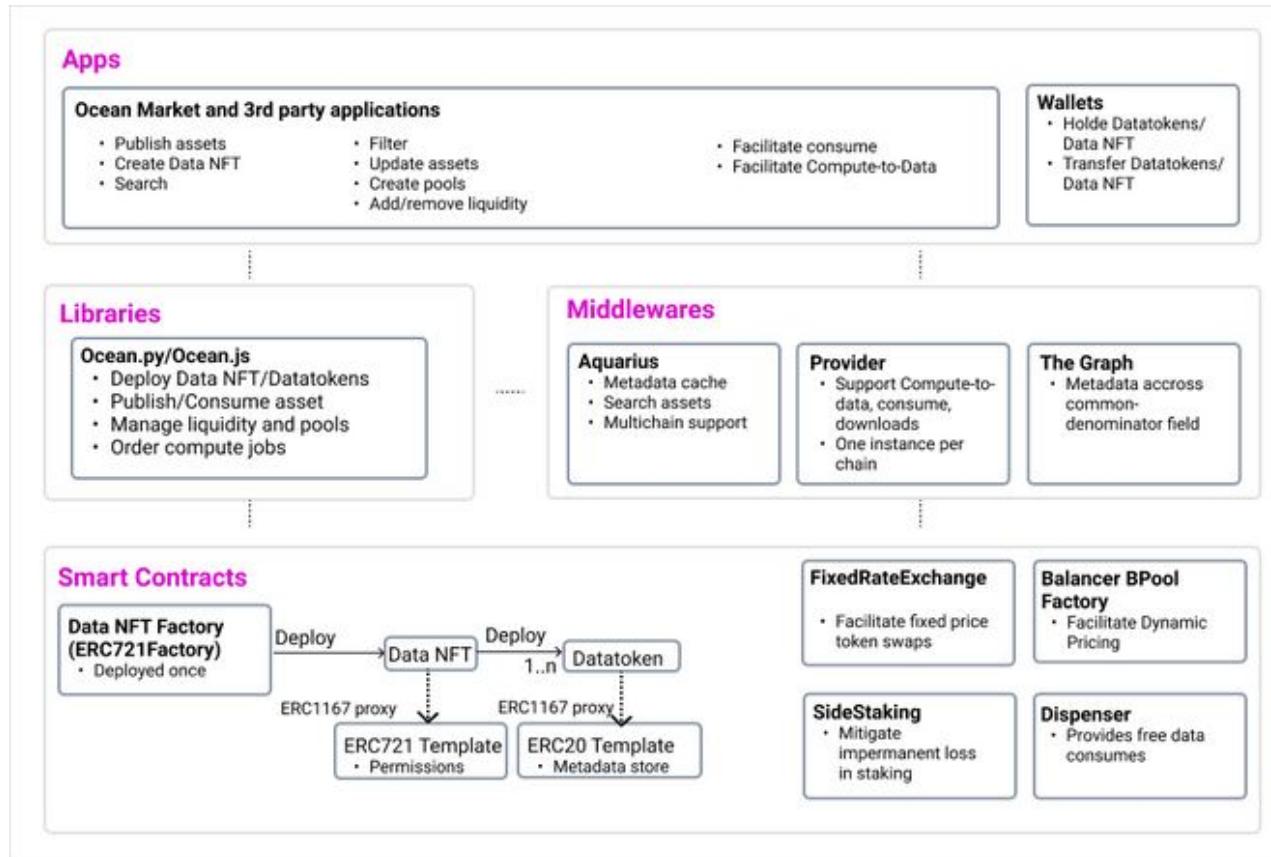
<here: Bob gets 1.0 datatokens>

```
# 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
)
```



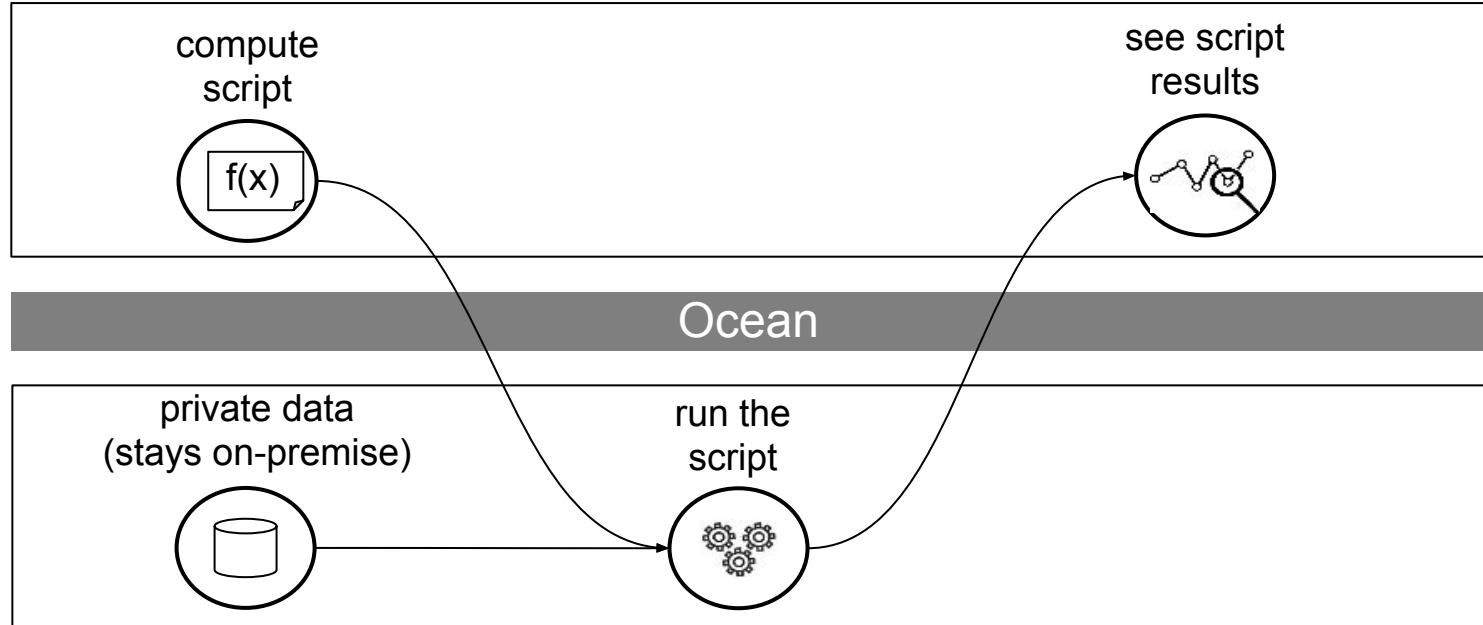
Detail: Ocean architecture





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 cre

[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](#)



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

3. Alice publishes algorithm

For this step, there are some prerequisites needed. If you want to replace the sample algorithm, you will need to do some dependency management. You can use one of the standard Ocean algorithms or 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=ocean.create_random_bytes(32))
ALG_datatoken.mint(alice_wallet.address, toWei(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",
        "script": "GPR.py"
    }
}
```

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 right asset
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
```

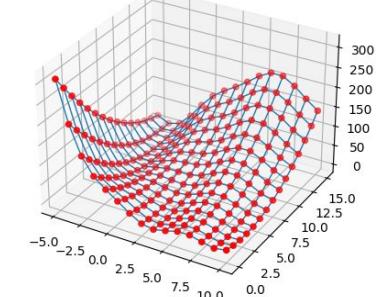
You can use the result however you like. For the purpose of this example, let's plot it.

```
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*t**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: 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_datatoken.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)
```

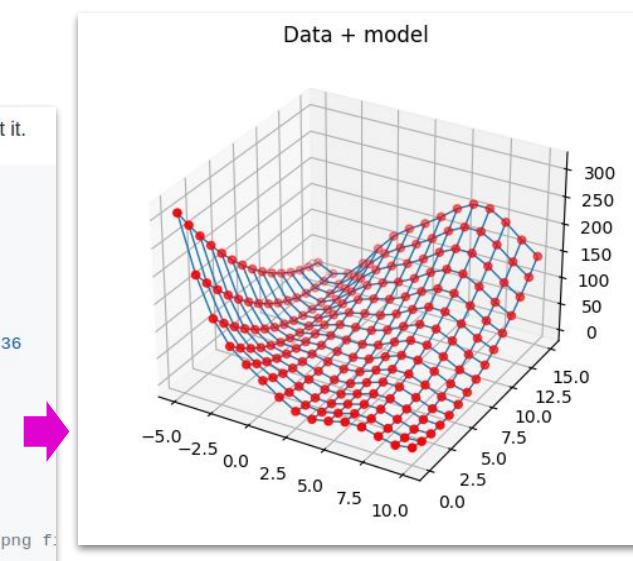
This will output the status of the current job. Here is a list of possible results: [Operator Service Status description](#).

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.*(. - 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
```



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 cre

[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](#)



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

3. Alice publishes algorithm

For this step, there are some prerequisites needed. If you want to replace the sample algorithm, you will need to do some dependency management. You can use one of the standard Ocean algorithms or 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.address, blob=ocean.create_random_bytes(32))
ALG_datatoken.mint(alice_wallet.address, toWei=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",
        "script": "GPR.py"
    }
}
```

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 right asset
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
```

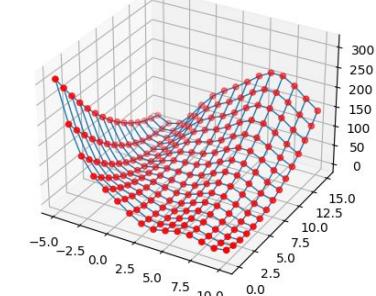
You can use the result however you like. For the purpose of this example, let's plot it.

```
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*t**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



Compute-to-Data In Ocean Market

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

The image displays four screenshots illustrating the Ocean Market's Compute-to-Data feature:

- Publish Page:** Shows the "Publish" section where users can highlight features of their data set or algorithm. It includes a note about beta status and links to the market, risks, and terms of use.
- Algorithm Detail Page:** Shows the "Random Forest Classifier v1.0" published by "OxAcca_1f83". It includes a description, tags ("random-forest", "classifier"), and data author information ("Raven Protocol & Ocean Protocol").
- Market Search Results:** Shows a search results page with 31 results, filtered by "ALGORITHMS". It lists items like "Apply pandas filter", "Logistic Regression v1.0", and "Random Forest Regressor v1.0".
- Job Details:** Shows a completed job titled "Job finished" for "Daily Fishing Effort (01.01.2020)". It includes details like the DID, creation and finish times, and the job ID.





Ocean Market:
Decentralized data
market for algorithms +
data

Ocean Market: Splash Page

The screenshot shows the Ocean Market splash page at market.oceanprotocol.com. The page features a large, stylized wavy graphic in the background, with a pink line on the left labeled "Bookmarks" and "Your bookmarks will appear here." and a grey line on the right labeled "Highest Liquidity". At the top, there is a navigation bar with links for "PUBLISH" and "PROFILE", a search bar, and a status message "Reconnecting...". The main title "Ocean Market" is displayed prominently, followed by the subtitle "A marketplace to find, publish and trade data sets in the Ocean Network." Below the title, three data set cards are shown:

- QUICRA-0**
DataUnion.app - Image & Annotation Vault
DataUnion.app
Notice This dataset and the software stack behind it are under constant de...
210.354 OCEAN POOL ETH
- LUMSTA-42**
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...
82,862.46 OCEAN POOL ETH
- EXCANE-93**
EVO/2MP/TRFC/DE/200K Weekly Collector
Evotegra GmbH
Evotegra - EVO/2MP/TRFC/DE/200K German Traffic Data for Machine Lear...
1,511.099 OCEAN POOL ETH



Ocean Market: Publish Flow, for an Initial Data Offering

The screenshot shows the first step of the publish flow, titled "Publish". It includes fields for "Title*", "Description*", "File*", "Sample file", and "Access Type*". A note at the top right says "PUBLISH". A yellow sidebar on the left provides beta status information.

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*
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.

Sample file
e.g. <https://file.com/samplefile.json>

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*

The screenshot shows the continuation of the publish flow. It includes fields for "Access Type*", "Datatoken Name & Symbol*", "Author*", "Tags", and "Terms & Conditions". A note at the top right says "PUBLISH". A yellow sidebar on the left provides beta status information.

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.

Datatoken Name & Symbol*
- C

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

Author*
e.g. Jelly McJellyfish

Give proper attribution for your data set.

Tags
e.g. logistics, ai

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 Commerze @ 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

SUBMIT **RESET FORM**

Example Data Asset, for Fixed Price

The screenshot shows the Ocean Market interface with a fixed price data asset listed.

Top Bar: Ocean Market (Beta) logo, PUBLISH, HISTORY, Get MetaMask, and settings icon.

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

Left Panel (Data Reservoir):

- Dataset Info:** Exceptional Whale Token – EXCWHA-70, Published by 0x98EA_16E4 – Etherscan.
- Logo:** DATA RESERVOIR (blue stylized 'dr').
- Description:** 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.
- Update Frequency:** Updated monthly.

Right Panel (Marketplace):

- Options:** USE, POOL, TRADE.
- Pool Details:** 2,639.166 OCEAN POOL (€1,211.94).
- Buy Button:** BUY.
- Note:** For using this data set, you will buy 1 EXCWHA-70 and immediately spend it back to the publisher and pool.
- Status:** No account connected. Please connect your Web3 wallet.

Bottom Left: 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

Bottom Right: Ocean logo.

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

The screenshot shows the Ocean Market interface. At the top, there's a navigation bar with the Ocean logo, "Ocean Market BETA", "PUBLISH", "HISTORY", a "Get MetaMask" button, and a settings icon.

The main content area displays a data asset titled "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 inquiries:

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

How it works.

On the right side, there's a "USE" tab selected, showing a "POOL" section with the following details:

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 shows the Ocean Market BETA interface. At the top, there is a navigation bar with the Ocean logo, "Ocean Market BETA", "PUBLISH", "HISTORY", a "Connect Wallet" button, and a settings icon.

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)
- Geo coverage: Worldwide

USE **POOL** **TRADE**

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 v3 interface at market.oceanprotocol.com. The top navigation bar includes links for 'Ocean Market v3', 'PUBLISH', and 'PROFILE'. A message at the top states: 'We are in beta. Please familiarize yourself with the [market](#), the [risks](#), and the [Terms of Use](#)'. The main title 'Ocean Market' is centered above a subtitle: 'A marketplace to find, publish and trade data sets in the Ocean Network'. On the left, there's a decorative graphic with wavy lines and text: 'Bookmarks' (with placeholder 'Your bookmarks will appear here.') and 'Highest Liquidity'. Three data set cards are displayed: 'QUICRA-0 DataUnion.app - Image & Annotation Vault' (692.868 OCEAN POOL), 'LUMSTA-42 Product Pages of 1'044'709 Products on Amazon.com (process...' (83,942.031 OCEAN POOL), and 'EXCANE-93 EVO/2MP/TRFC/DE/200K Weekly Collector' (1,523.022 OCEAN POOL). The right side features a 'Networks' panel with two sections: 'Main' (ETH, Polygon, BSC) and 'Test' (ETH Ropsten, ETH Rinkeby, Polygon Mumbai, Moonbase Alpha, GAIA-X Testnet). A large pink arrow points from the text 'Ocean is multi-chain' in the image title down to the 'Main' networks section in the screenshot. The bottom right corner has the 'ocean' logo.

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

Ocean Market v3 PUBLISH PROFILE

Search... OxF5dc...5497

Ocean Market

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

Bookmarks
Your bookmarks will appear here.

Highest Liquidity

DATA SET

DATA SET

DATA SET

Networks

Switch the data source for the interface.

Main

⚡ ETH

☁ Polygon

⚡ BSC

Test

⚡ ETH Ropsten

⚡ ETH Rinkeby

☁ Polygon Mumbai

🌐 Moonbase Alpha

⚡ GAIA-X Testnet

692.868 OCEAN POOL ⚡ ETH

83,942.031 OCEAN POOL ⚡ ETH

1,523.022 OCEAN POOL ⚡ ETH

ocean

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



Ocean Protocol @oceanprotocol · Sep 22

TECHNICAL UPDATE | Fine-Grained Permissions have now been launched on the Ocean Market that will offer the enterprises and other users more precise ways to specify and manage access. This new update addresses issues of access control along two main levels:

```
graph TD; User((User)) -- "1. User requests access to browse/consume/publish" --> Keycloak[Keycloak]; Keycloak -- "3. Request role for Ethereum Address" --> RBAC(RBAC Server); RBAC -- "4. Response: User role" --> Market[Market]; RBAC -- "2. Request permission" --> Market; Market -- "5. Response: true / false" --> User; Market -- "5. Market access allowed or denied" --> User;
```

The diagram illustrates the process of requesting fine-grained permissions. A user initiates a request to a Keycloak instance. Keycloak then sends a request to an RBAC Server for a specific role based on the user's Ethereum address. The RBAC Server returns the user's role to Keycloak. Finally, Keycloak sends a request to the Market to check if the user has the required permission. The Market returns a response indicating whether access is granted or denied, which is then communicated back to the user.

5 41 136

The screenshot shows the Ocean Market interface with a focus on fine-grained permissions. It displays a 'Sample Files' section for a dataset titled 'LoRaWAN System Token - LODVHS-SI.r' published by 'OCEAN-2003 - Autopilot on Blue' 7 days ago. The 'Licensing' section indicates the files are under the GNU GPL license. Below this, there are sections for 'DATA AUTHOR' (zombie) and 'CLOUD' (OCEAN-2003 - Autopilot on Blue). The 'DID' section shows the DID as did:ip:15C8Ba26275b04A6f68115C3319008e792000097. The 'METADATA HISTORY' section shows it was published 7 days ago. At the bottom, there are sections for 'Edit ETH Address' and 'Disable Consumption'. A green arrow points from the text 'set. Updating these settings will create an NFT in your wallet.' to the 'Edit ETH Address' button.



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.



ocean

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>

