Contents

[Links to Pull Requests 1](#_Toc125987504)

[Introduction 1](#_Toc125987505)

[Flow Diagram 2](#_Toc125987506)

[META NFT 2](#_Toc125987507)

[Claim NFT 3](#_Toc125987508)

[Algorithm/Claim/Dataset Comments section 4](#_Toc125987509)

[Claim Logs 4](#_Toc125987510)

[Code Change Summary 5](#_Toc125987511)

[UI Changes (Market) 5](#_Toc125987512)

[Provider 11](#_Toc125987513)

[Pod Configuration 12](#_Toc125987514)

[Operator Engine 13](#_Toc125987515)

# Links to Pull Requests

https://github.com/oceanprotocol/market/pull/1872

https://github.com/oceanprotocol/provider/pull/594

https://github.com/oceanprotocol/operator-engine/pull/74

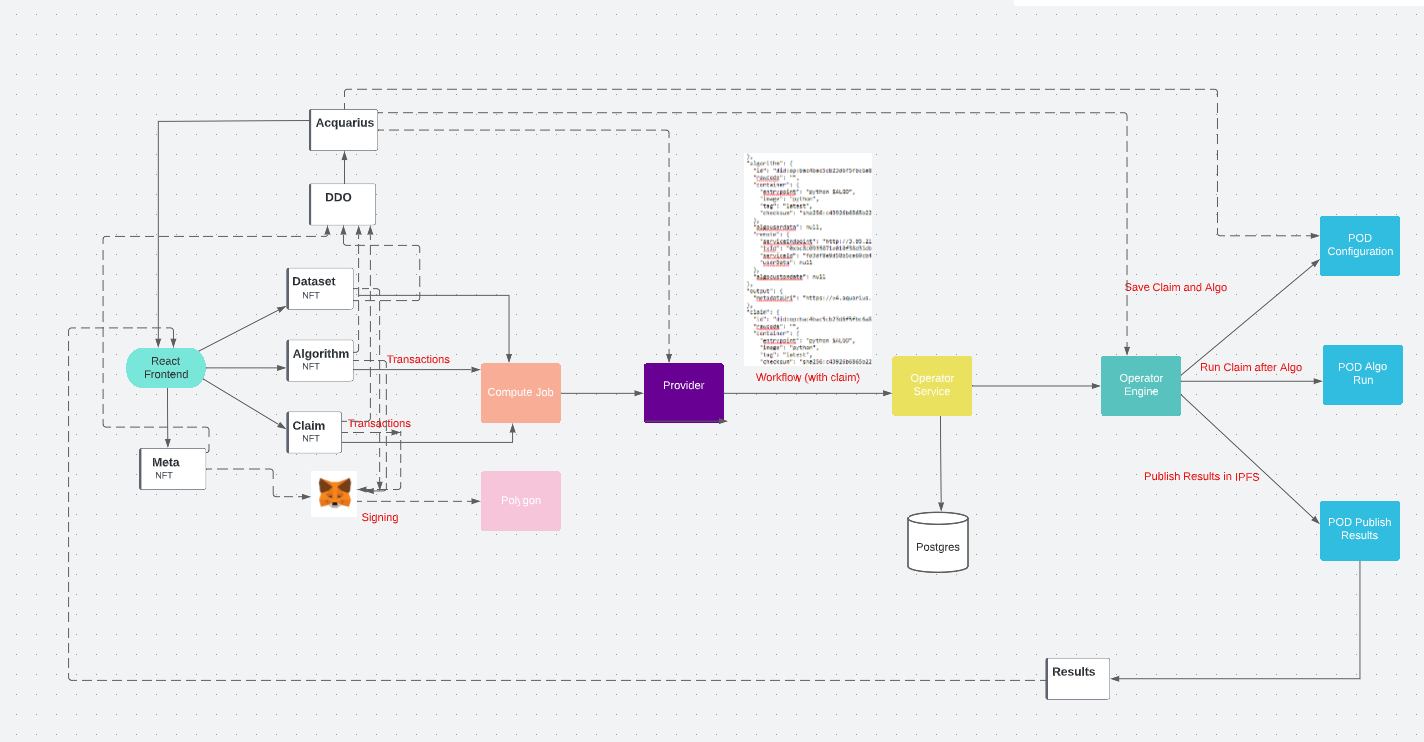
https://github.com/oceanprotocol/pod-configuration/pull/49

# Introduction

Claims are algorithms that are run after execution of algorithm on a dataset. This is useful for enterprise users who can run a claim to perform some specific checks post algorithm execution. In order to store the history of a claim execution a metadata NFT is introduced in Ocean Protocol. The metadata NFT has a type “meta” and it stores the detail of every claim executed by a user (and in future can also be used to store comments). Any claim execution is a meta data update of the meta NFT. In order to retrieve the comment or claim executed on a particular algorithm or dataset we retrieve all available metadata NFT and get details of the pertaining algo or dataset.   
The below document list changes in all components for this.

# Flow Diagram

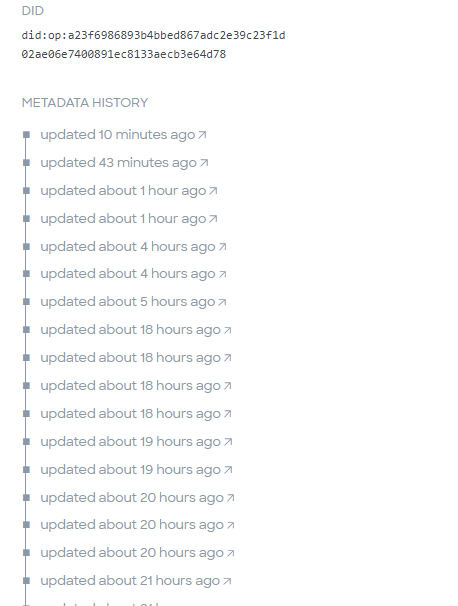
https://lucid.app/lucidspark/150850b9-d8a0-4332-ac7d-a2ac914d45ac/edit?invitationId=inv\_086805f3-55a4-4154-acf0-da07fcb3fa0b#



### META NFT

Meta NFT stores all comments and claims. The meta NFT is a DDO stored on chain with the type ‘meta’. There is a transaction in meta NFT everytime a claim is done. There can be only one meta NFT per user.

Meta NFT transactions (updated every time any claim is run)



### Claim NFT

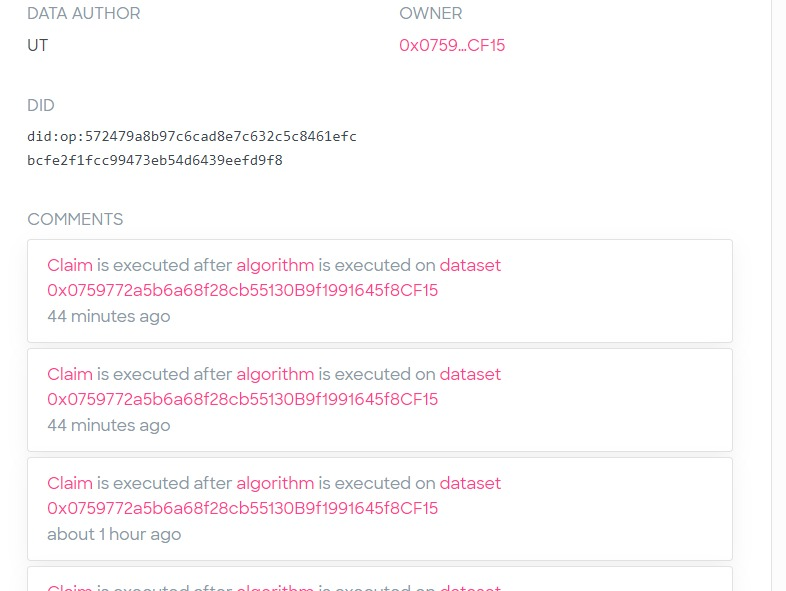
Claim is a special algorithm that is executed after an algo executes over a dataset

There will be a single META token for each chain/user but there can be any number of claims.

Claims have type **CLAIM**

### Algorithm/Claim/Dataset Comments section

The Algorithms, datasets and claims now have a comments section which are pulled from all available meta NFTs



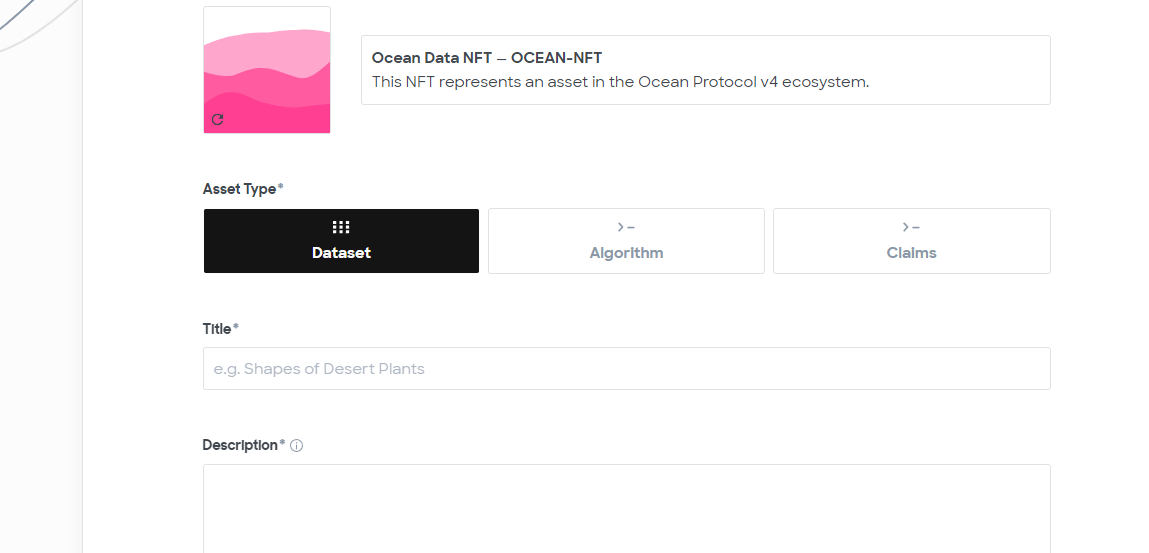
### Claim Logs

Claim logs are appended in the end in algorithm logs and pushed to IPFS

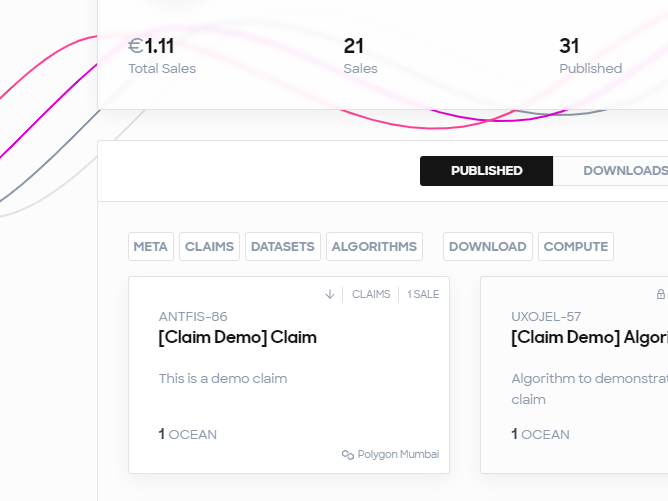
# Code Change Summary

## UI Changes (Market)

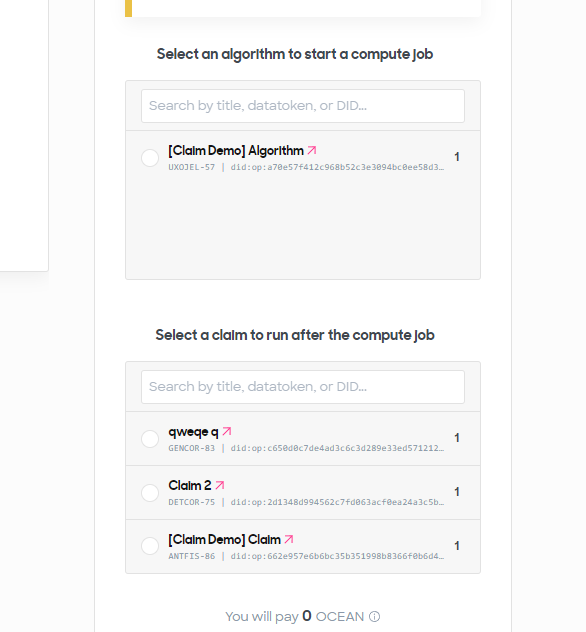
* Changes to create the meta NFT. This is currently done when a new dataset is published. We publish a meta NFT with type ‘meta’ with all other settings similar to the dataset. The description of this NFT is important as that’s where the comments and claims are stored.
* Changes to publish a claim. A claim is just an algorithm but with a type ‘Claims’. So there is a new tab added in UI beside algorithm to add claims.



* Changes to filters to display meta and claims



* Changes to show claims while running dataset



Only the owners claims are showed.

* Once the compute job is started the meta NFT is retrieved and a claim string is appended in the description of meta NFT in the below format.

{

  "metadata":

    [

  {"TemplateType": "ClaimHistory",

        "Claim": "did",

        "Dataset": "did",

        "Algorithm": "did",

        "Comment": "Claim {0} is executed after algo {1} is executed om dataset {2}"

      },

      {"TemplateType": "ClaimHistory",

        "Claim": "did",

        "Dataset": "did",

        "Algorithm": "did",

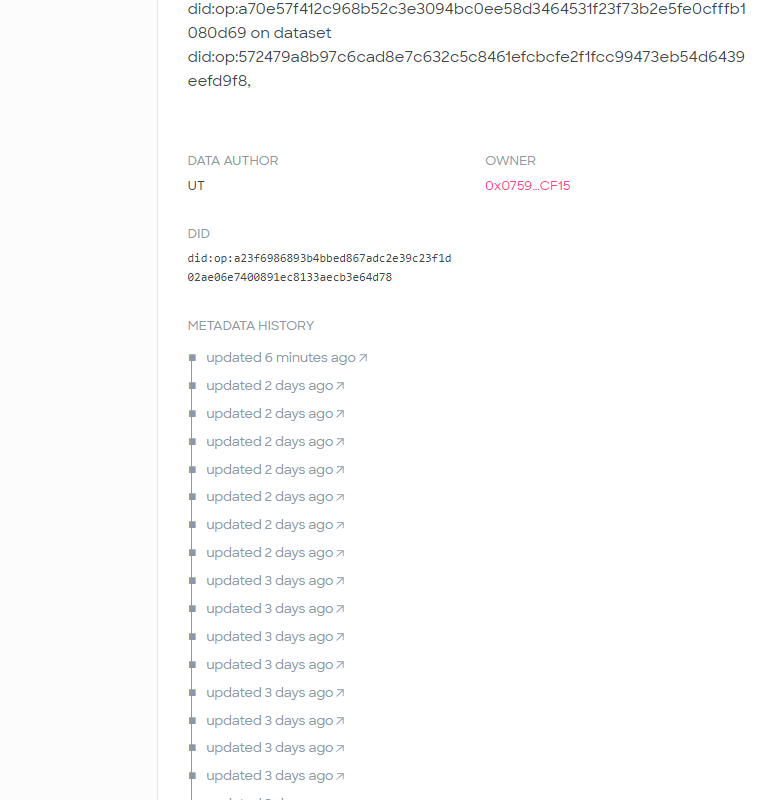
        "Comment": "Claim {0} is executed after algo {1} is executed om dataset {2}"

      }

    ]

  }

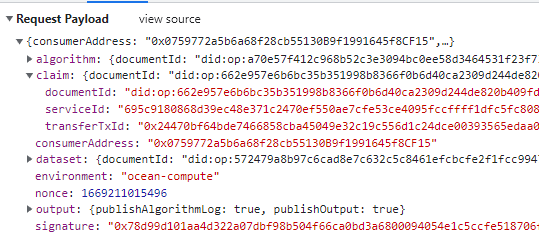
This is also an on chain transaction



* The claim id is then initialized in provider as the algorithm during initializecompute and sent in the compute

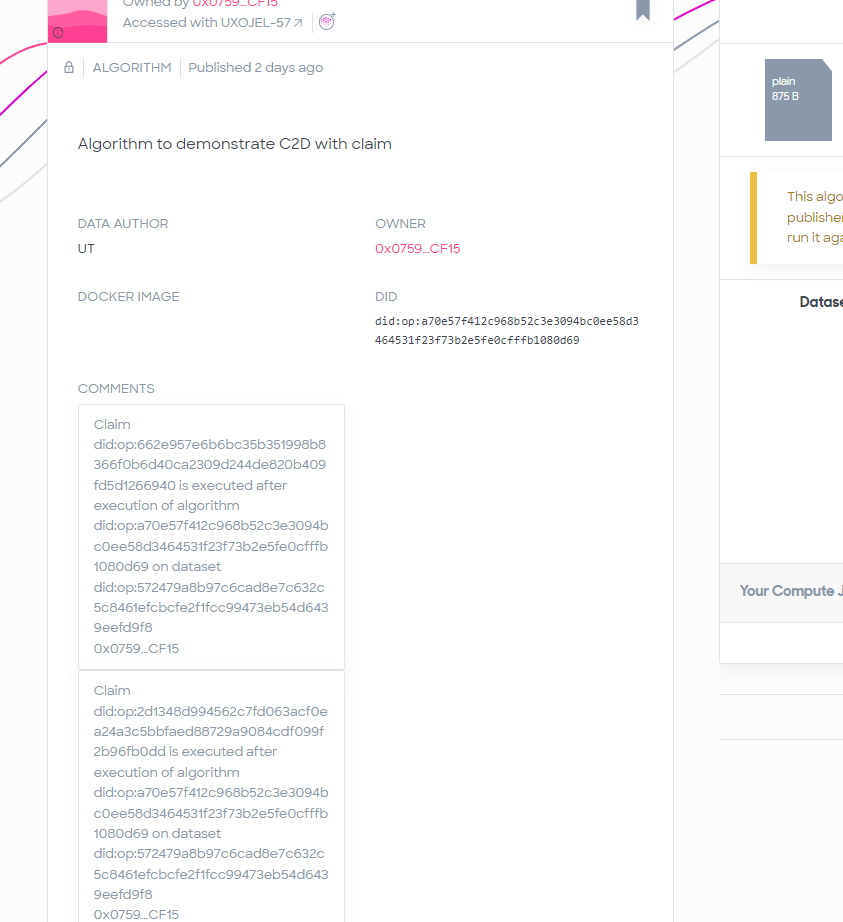


* The compute endpoint is changed to accept claim in the payload



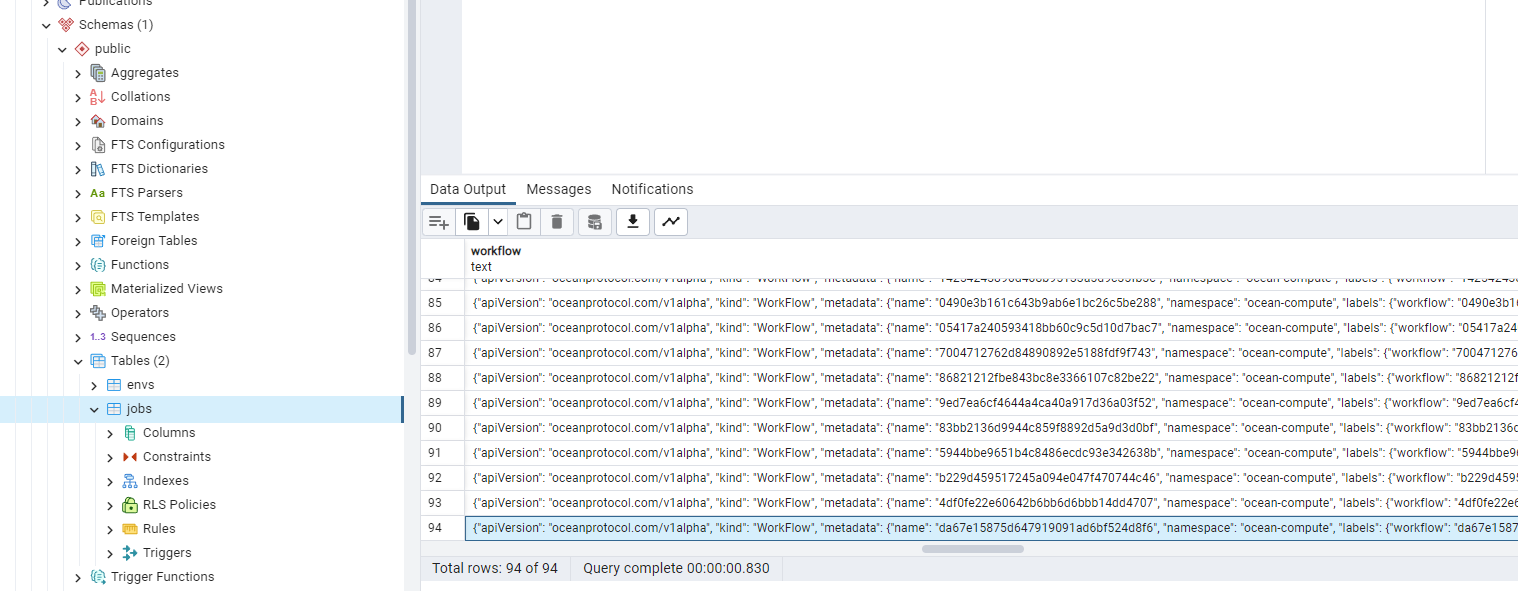
* In the algorithm page –

1. All meta NFTs are retrieved
2. Their comments are parsed
3. The matching comments are shown as below –



## Provider

The compute endpoint is changed in provider to accept the claim payload. It then validates it as it validates an algorithm and creates the workflow with claim appended. Database snapshot below



Part of workflow json with claim appended below algorithm–



## Pod Configuration

Pod configuration is changed to read the claim section and save the claim in /data/transformations/claim similar to algorithms saved in /data/transformations/algorithm

## Operator Engine

Operator engine is changed to pass on the claim in the entry point of algorithm pod YAML (command section)

