Guide on how to organise the metadata of your application into an excel sheet to make it runnable

Compiled by University of Westminster for DIGITbrain

20/04/2022

Jozsef Kovacs on behalf of
James Deslauriers, Resmi Arjun, Hamed Hamzeh
and Tamas Kiss

Goal of this presentation

- To guide you through the steps on **how to organise/fill the metadata** of your assets
- To **produce an excel sheet** which then
- Can be converted into a **runnable application**

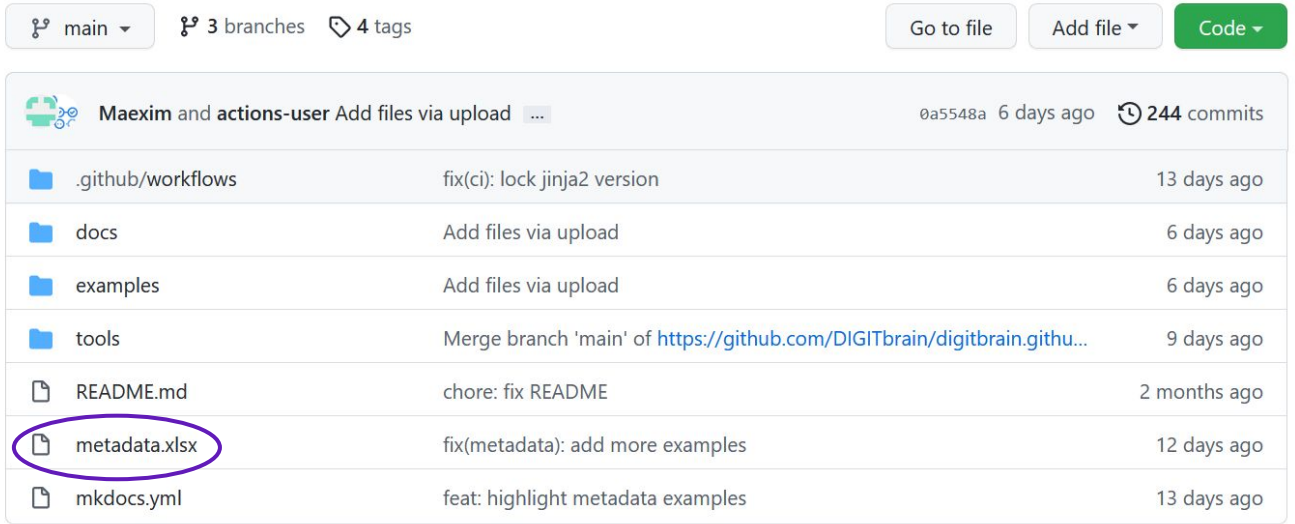
- Therefore this guide **focuses on those key value pairs** of the asset metadata which are **required to produce** a valid **set of asset metadata convertable** into a runnable application (DMA)

- **Important:** whenever a cell (belonging to a key) value requires structured info instead of a single string,
  you must use **json** format!

- NB: The procedure described here is a temporal solution/work-around which is going to be replaced by the Publishing Interface in the future.

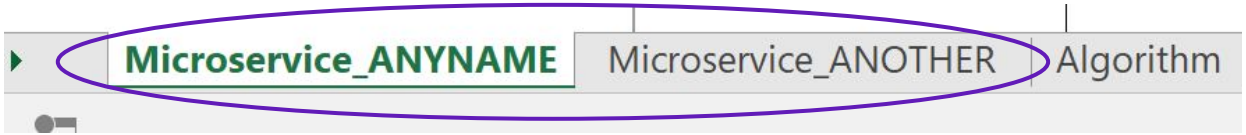
Step 0: download the metadata template

- make sure you download the latest version of the metadata Excel file from https://github.com/DIGITbrain/digitbrain.github.io



Step 1: create microservice asset placeholders

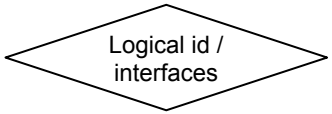
1. make as many copies of the microservice sheets as you will need to implement your example and name each sheet **"Microservice_<name>"**.



2. IDs need to be generated manually for each Microservice asset. The IDs will be referred to/referenced in other assets. Please fill in the key called "id". We suggest naming the ID as follows:
   - "**MSID_<name>**" The name string has to be chosen uniquely.

Microservice Metadata

Key	Subkey	Type	Example Value	Comment	Condition	Values
id		String	"MSID_MYMS_A"	unique DIGITbrain reference	auto	"MSID_FLOWER"



consists of

refers to

MS

MS

Step 2: define your microservice(s)

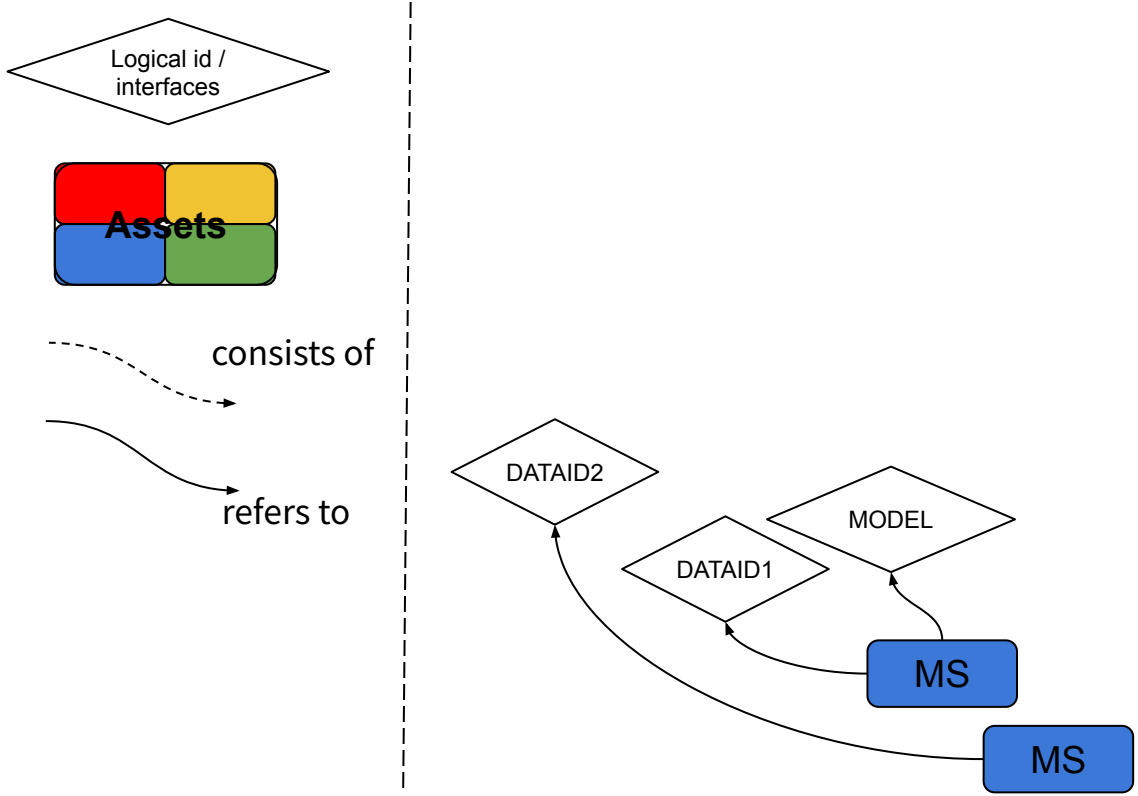
- For each of your microservice the following fields in the **"Microservice"** sheet are mandatory to generate a runnable application:
- specify **"deploymentFormat"** as for example **"docker-compose"**
- specify **"deploymentData"** with a compose file that is able to execute a single container.
  (do not use compose having multiple containers in it!)

Step 2: define your microservice(s)

```
{
  "version": "3.7",
  "services": {
    "mycontainername": {
      "image": "dbs-container-repo.emgora.eu/mydockerimage",
      "entrypoint": "/bin/sh -c",
      "command": "/path/to/myexecutable --data1 {{DATAONE.DATA_URI}}
      --data2 {{DATATWO.DATA_URI}} --model {{MODEL.REPOSITORY_URI}}"
    }
  }
}
```

{{DATAONE.DATA_URI}} is a placeholder that will be substituted with the value of the "DATA_URI" parameter defined in your "DATA" asset/sheet in the excel that has been bound with **DATAONE logical id** defined later. Using this mechanism, any parameter inside the DATA asset can be referred.

{{MODEL.REPOSITORY_URI}} is a placeholder that will be substituted with the value of the "REPOSITORY_URI" parameter defined in your "MODEL" asset/sheet in the excel (Please, note there can only be one model in an application!)


Logical id / interfaces
Assets
consists of
refers to
DATAID2
MODEL
DATAID1
MS
MS

Step 3: create an algorithm asset

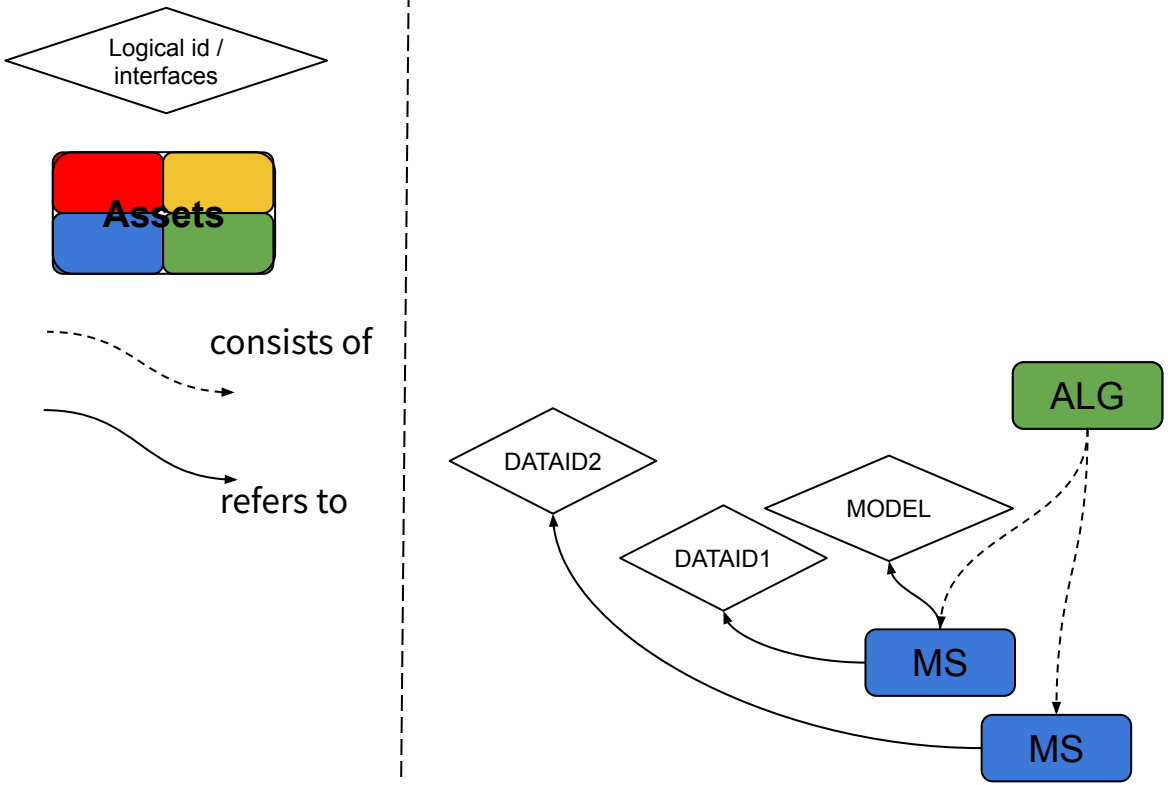
1. Please fill in the key called **"id"**. We suggest naming the ID as follows:
   - **"ALGID_<name>"** for algorithm asset.

Algorithm Metadata

Key	Subkey	Type	Example Value	Comment	Condition	Values
id		String	"ALGID_MYALG"	a unique id to identify this asset	auto	"ALGID_FLOWER"

2. In the **Algorithm sheet**, specify the **"listOfMicroservices"** key by listing your Microservice(s) to be contained in the Algorithm. List them by their IDs, (use json format!), e.g. **"["MSID_ONE", "MSID_TWO"]"** or e.g. **["MSID_FLOWER"]**

date		Date	06.04.2021	the creation data	auto	
version		String	"1.0"	the version	mandatory	"1.0.0"
listOfMicroservices		List[String]	["MSID_MYMS_A", "MSID_MYMS_B"]	a list of Microservice Asset IDs, which are contained in the algorithm	mandatory	["MSID_FLOWER"]
deploymentMapping		Map[String, String]	{"MSID_MYMS_A": "HOSTID_MYHOST_A", "MSID_MYMS_B": "HOSTID_MYHOST_A"}	a mapping specifying which microservice should run on which host. By default each microservice is assigned a respective host, but this behaviour is not always ideal (eg. when two or more Microservices may need to run on the same host)	mandatory	{"MSID_FLOWER":"HOSTID_FLOWER"}


Logical id / interfaces
Assets
consists of
refers to
ALG
DATAID2
MODEL
DATAID1
MS
MS

Step 4: bind host id(s) to microservice(s)

On the **Algorithm** sheet, we have to set which microservice should run on which host. For this, the **deploymentMapping** must be specified with microservice-host pairs using JSON style. Examples:

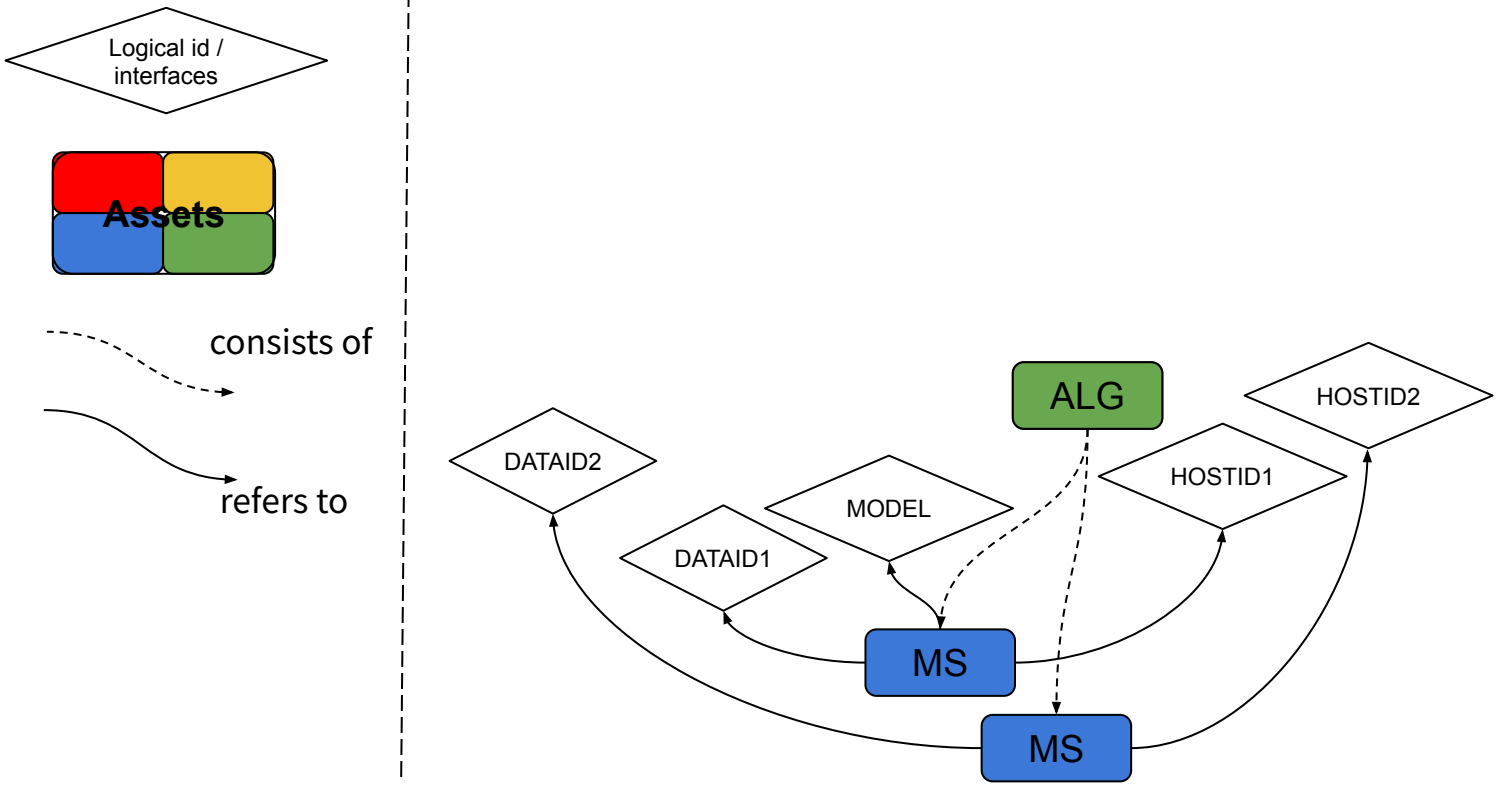
1. Two microservices should run on two separate hosts:

```
{
  "MSID_ONE": "ID_OF_YOUR_HOST",
  "MSID_TWO": "ID_OF_ANOTHER_HOST"
}
```

2. Two microservices should run on the same host:

```
{
  "MSID_ONE": "ID_OF_A_HOST",
  "MSID_TWO": "ID_OF_A_HOST"
}
```

Algorithm							
	listOfMicroservices		List[String]	["MSID_MYMS_A", "MSID_MYMS_B"]	a list of Microservice Asset IDs, which are contained in the algorithm	mandatory	["MSID_FLOWER"]
	deploymentMapping		Map[String, String]	{"MSID_MYMS_A": "HOSTID_MYHOST_A", "MSID_MYMS_B": "HOSTID_MYHOST_A"}	a mapping specifying which microservice should run on which host. By default each microservice is assigned a respective host, but this behaviour is not always ideal (eg. when two or more Microservices may need to run on the same host)	mandatory	{"MSID_FLOWER":"HOSTID_FLOWER"}


Logical id / interfaces
Assets
consists of
refers to
ALG
HOSTID2
DATAID2
HOSTID1
MODEL
DATAID1
MS
MS

Step 5: specify the Model asset

1. Please fill in the key called **"id"**. We suggest naming the ID as follows:
   - **"MODELID_<name>"** for Model asset.
2. At least those **keys** which are referred by the microservice asset need to be specified

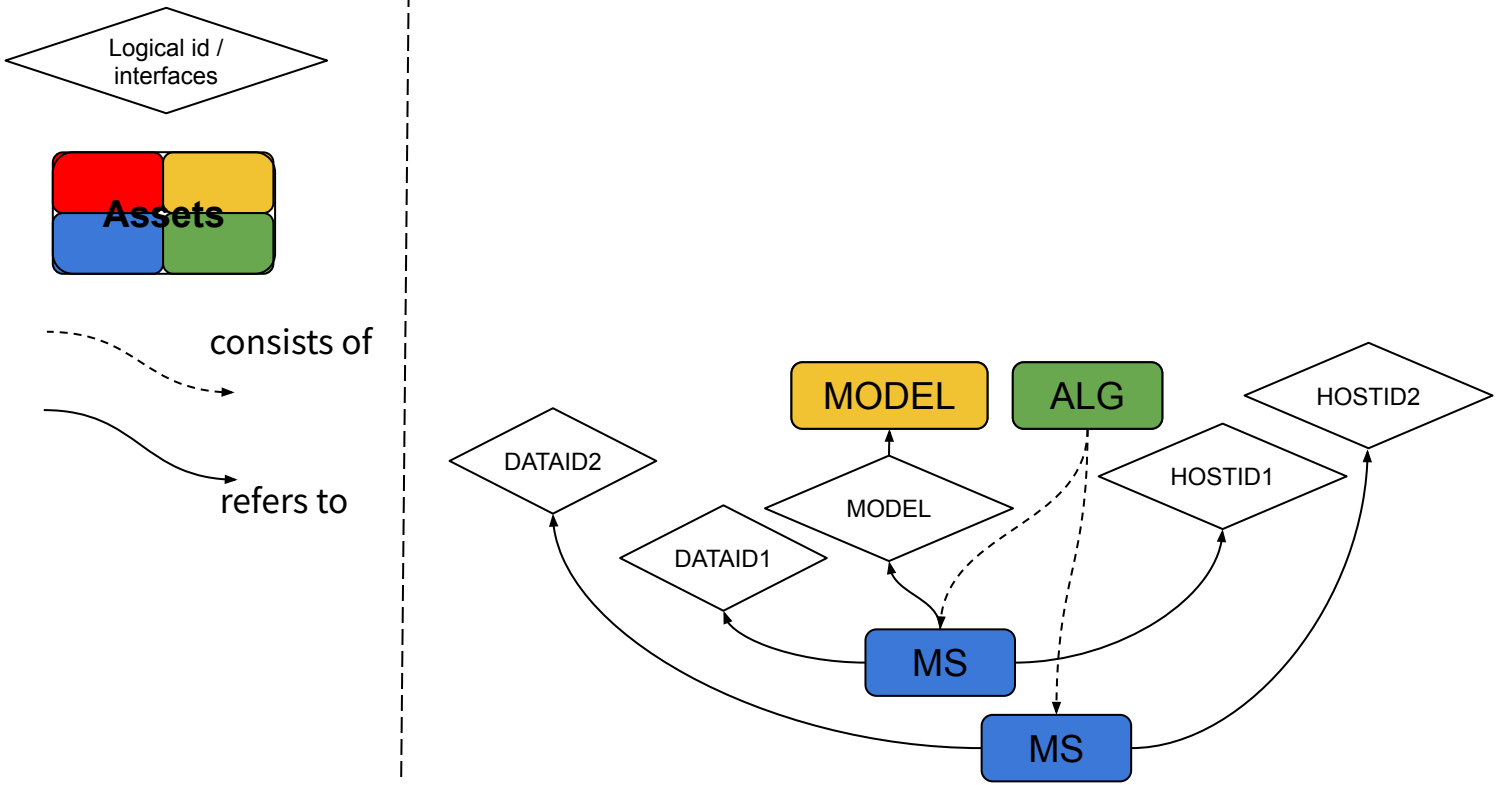
   In "Model" sheet these keys are the following: **“REPOSITORY_URI, “PATH”, and “FILENAME”**. The **value** you assign/give to this key will be **substituted into a placeholder** defined in the **deploymentData** of one of your Microservices.

   Note: Your Microservice is responsible to download the Model (or use rclone to do it for you)

				specific algorithms) - The original term Path was proposed to be changed (SAD)		
Fidelity		number		Error of the model’s prediction	optional	SavedModel
REPOSITORY_URI		URI	"https://www.myrepo.com"	Where the model file is stored (usually the DigitBrain certified external model repository). The path and model filename are not provided via this field.	mandatory	"s3://dbrain-s3-test"
PATH		string	"input/models"	Path to the model file in the specified repository, not including the filename itself.	mandatory	"dfki/flower/model"
FILENAME		string	"mymodel.pb"	Name of the model file at the given path within the given repo, with a file extension if it exists.	mandatory	"dfki_ml_flower-inference.zip"
State_depend		bool		stateful -> 1, stateless -> 0	optional	

Remember how model parameters were referred:

```
"command": "/path/to/myexecutable --data1 {{DATAONE.DATA_URI}}
--data2 {{DATATWO.DATA_URI}} --model {{MODEL.REPOSITORY_URI}}",
```

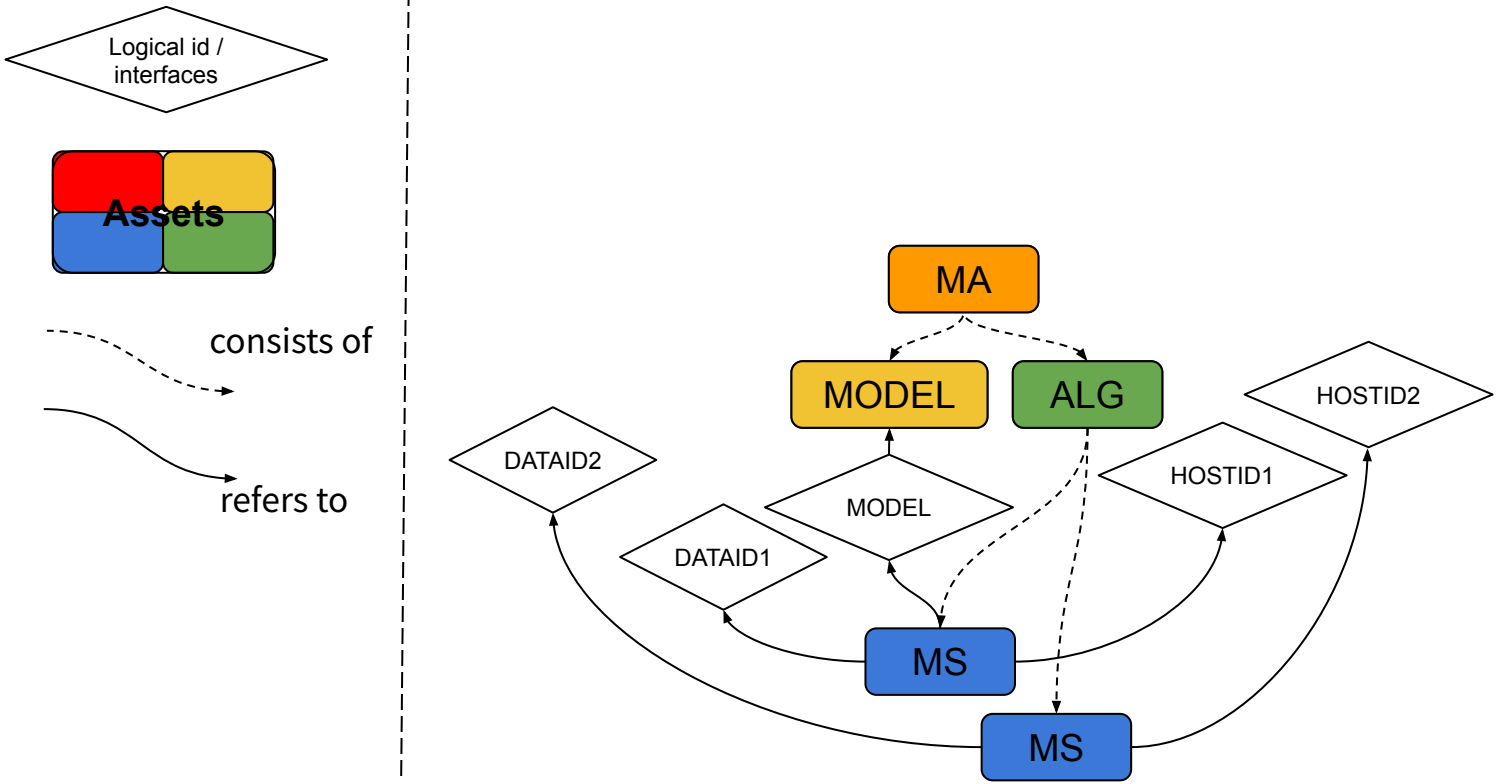

Logical id / interfaces
Assets
consists of
refers to
MODEL
ALG
HOSTID2
DATAID2
MODEL
HOSTID1
DATAID1
MS
MS

Step 6: define your MA pair

1. In the MA Pair sheet, please fill in the key called **"ID"**.
   - We suggest naming the MA pair ID as follows: **"MAID_<name>"**
2. specify the key **"A Asset"** to refer to your Algorithm asset by entering its ID **("ALGID_<name>")**
3. specify the key **"M Asset"** to refer to your Model asset by entering its ID **("MODELID_<name>")**

MA Pair Metadata

Key	Type	Comment	Condition	Values
ID	UUID	Unique identifier for the MA Pair	auto	"MAID_CAELIA"
Created at	ISO 8601	Date of creation	auto	
Version	Integer	Version number of the MA Pair		"1.0.0"
Author	UUID	Identifier of the Author of the MA Pair (NB: Entity for	auto	
Licensor	UUID	Identifier of the Legal Entity licensing the MA Pair (NB: Entity for Licensor is referenced)	auto	
Derivation	UUID	In case of derivation, references to parent / child	auto	
Name	Text	Short name to identify the MA Pair	mandatory	"ROM-Modelling"
Scope	Text	Short description of the scope of the MA Pair (human readable)	mandatory	"Physically based simulation ROM evaluation for given input values"
IP Family	UUID	Identifier of the IP Family the MA Pair is valid for	mandatory	"IP_Familiy_ID"
Namespace	UUID?	Context to interpret the associated information	optional	
M Asset	UUID	Identifier of the Model Asset associated to the MA Pair (NB: The corresponding model file is indicated in Model metadata (first tab) as Model_URI and the corresponding zip file needs to be served to the Algorithm to evaluate the Model by the Microservice (MODEL_FILENAME))	mandatory	"MODID_CAELIA"
A Asset	UUID	Identifiers of the Algorithm Asset associated to the MA	mandatory	"ALGID_CAELIA"


Logical id / interfaces
Assets
consists of
refers to
MA
MODEL
ALG
HOSTID2
DATAID2
HOSTID1
MODEL
DATAID1
MS
MS

Step 7: create your DMA, assign the MA to it

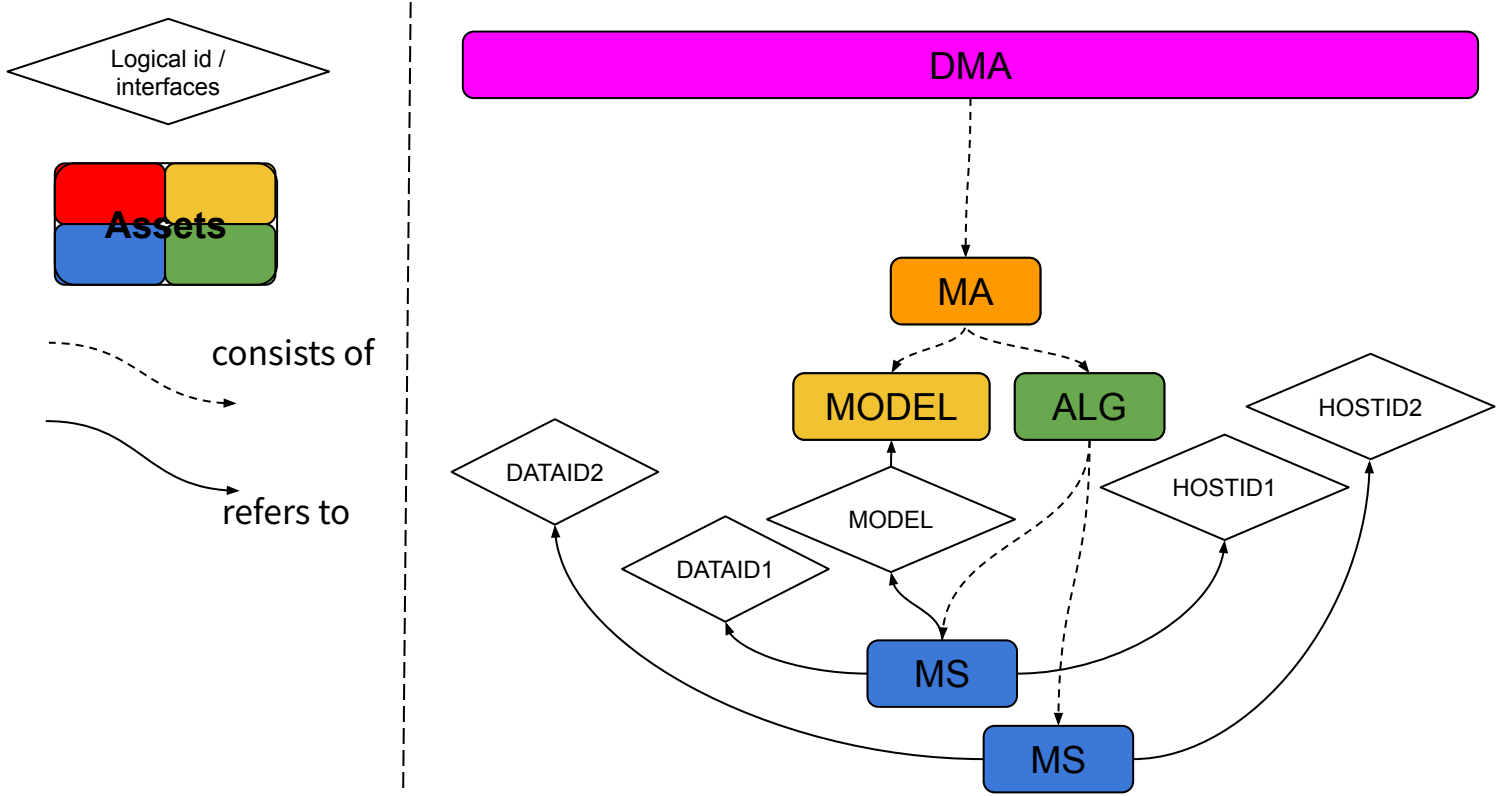
1. Please fill in the key called **"ID"**.
    - We suggest naming the ID as follows: **"DMAID_<name>"** for DMA.

2. In the **DMA sheet**, specify the **"MA Pair"** key to refer to your MA Pair by entering the ID **("MAID_<name>")**

DMA Tuple Metadata

Key	Type	Comment	Condition	Values
ID	ID / URI	Unique identifier for the DMA Tuple	auto	"DMAID_CAELIA"
Created at	ISO 8601	Date of creation	auto	
Version	Integer	Version number of the DMA Tuple		
Author	ID / URI	Identifier of the Author of the MA Pair (NB: Entity for author is referenced)	auto	
Licensor	ID / URI	Identifier of the Legal Entity licensing the the MA Pair (NB: Entity for Licensor is referenced)	auto	
Derivation	IDs / URIs	In case of derivation, references to parent / child	auto	
Name	Text	Short name to identify the DMA Tuple	mandatory	"ROM-Modelling hard-coded input"
Scope	Text	Short description of the scope of the DMA Tuple (human readable)	mandatory	"Physically based simulation ROM evaluation for given keyboard input values"
IP Instance	ID / URI	Identifier of the IP Instance the DMA Tuple is valid for (NB: Entity for IP Instance is referenced)	mandatory	"ID_IP_Instance"
Namespace	ID / URI?	Context to interpret the associated information	optional	
MA Pair	ID / URI	Identifier of the MA Pair associated to the DMA Tuple	mandatory	"MAID_CAELIA"
Schedule	Dates	Days and hours the DMA Tuple will be active (optional)	optional	
Payload	String	User-defined key-value pairs: JSON string with additional	optional	{'injectionMoulds': 'Model

orithm | Model | Data | MA Pair | **DMA Tuple** | Supporting Metadata | Deployment | DataAssetsMapping | ConfigurationData | Data Resources | Parameters | Metrics


Logical id / interfaces
Assets
consists of
refers to
DMA
MA
MODEL
ALG
DATAID2
MODEL
DATAID1
HOSTID2
HOSTID1
MS
MS

Step 8: create Data assets

1. make as many copies of the data sheet as you will need to implement your example and name the sheets accordingly, e.g. **"Data_anyname"**.

2. Please fill in the key called **"ID"**.
    - We suggest naming the ID as follows: **"DATAID_<name>"** for data assets.

Step 9: specify the keys of your Data assets

- In the **"Data"** sheet(s) those keys need to be filled that are referenced by a Microservice: e.g. **DATA_URI** .
- The **value** you enter/give to this key will be **substituted into a placeholder** defined in the **deploymentData** of one of your Microservices.

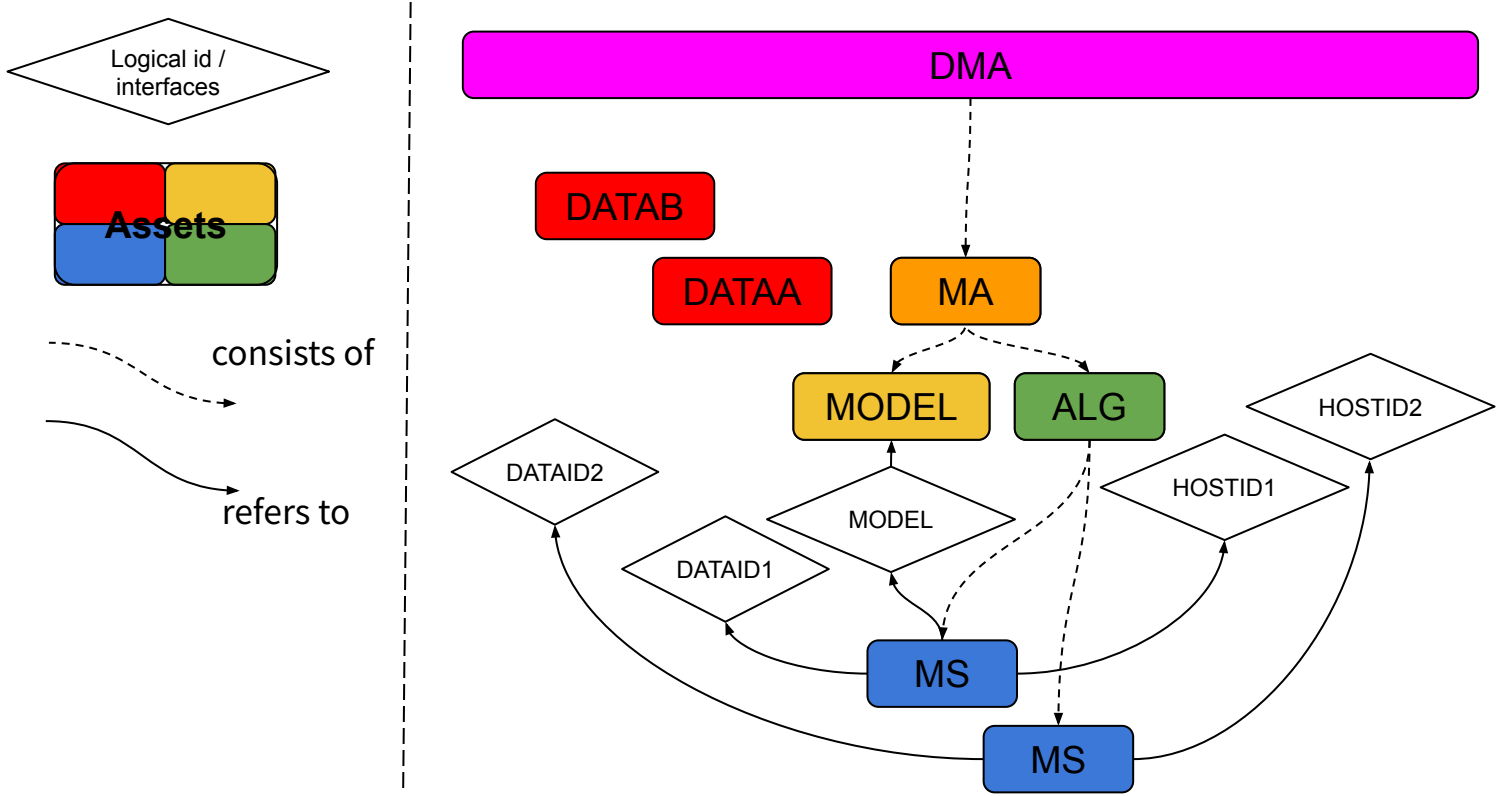
DIGITBRAIN

Data Metadata

Concept	Key	Type	Comment	Condition	Values
Data access specification					
	DATA_URI	URI	Accessibility of the data resource, including host, port information, protocol, and other fields (path is protocol dependent, can be a topic name). GUI may show host, port, path separately. Hidden at search. Format: scheme://host:port/path. Pseudo vars: DATA_PROTOCOL, DATA_HOST, DATA_PORT, DATA_PATH, DATA_QUERY, DATA_FRAGMENT.	mandatory (WP6 open)	"https://github.com/ResmiArjun/itainnova/raw/main/input/input.csv"

Remember how data parameters were referred:

```
"command": "/path/to/myexecutable --data1 {{DATAONE.DATA_URI}}
--data2 {{DATATWO.DATA_URI}} --model {{MODEL.REPOSITORY_URI}}",
```


Logical id / interfaces
Assets
consists of
refers to
DMA
DATAB
DATAA
MA
MODEL
ALG
HOSTID2
DATAID2
HOSTID1
MODEL
DATAID1
MS
MS

Step 10: link Data asset(s) to Microservice(s)

- In the **"DMA Tuple"** sheet please, specify the value for key **"dataAssetsMapping"** as follows (use **JSON** format!):

  Example:

  {"MSID_ONE": { "DATAONE": "DATAID_ANYNAME1" }}

- **MSID_ONE** is the id of the microservice that will use the values specified in the Data asset.
- **DATAONE** is a logical id (you can define it freely) which binds the Microservice and Data asset together and which can be referred to in the deploymentData of a Microservice.
- **DATAID_ANYNAME1** is the ID of Data asset defined in your excel sheet

- In case your Microservice requires more than one data asset, use the following structure (use JSON format!)

  {"MSID_ONE": { "DATAONE": "DATAID_ANYNAME2",

  "DATATWO": "DATAID_ANYNAME2" },

  "MSID_TWO": { "MYDATA": "DATAID_ANYNAME3" }}

Step 10: link Data asset(s) to Microservice(s) (2) (examples)

DIGITBRAIN

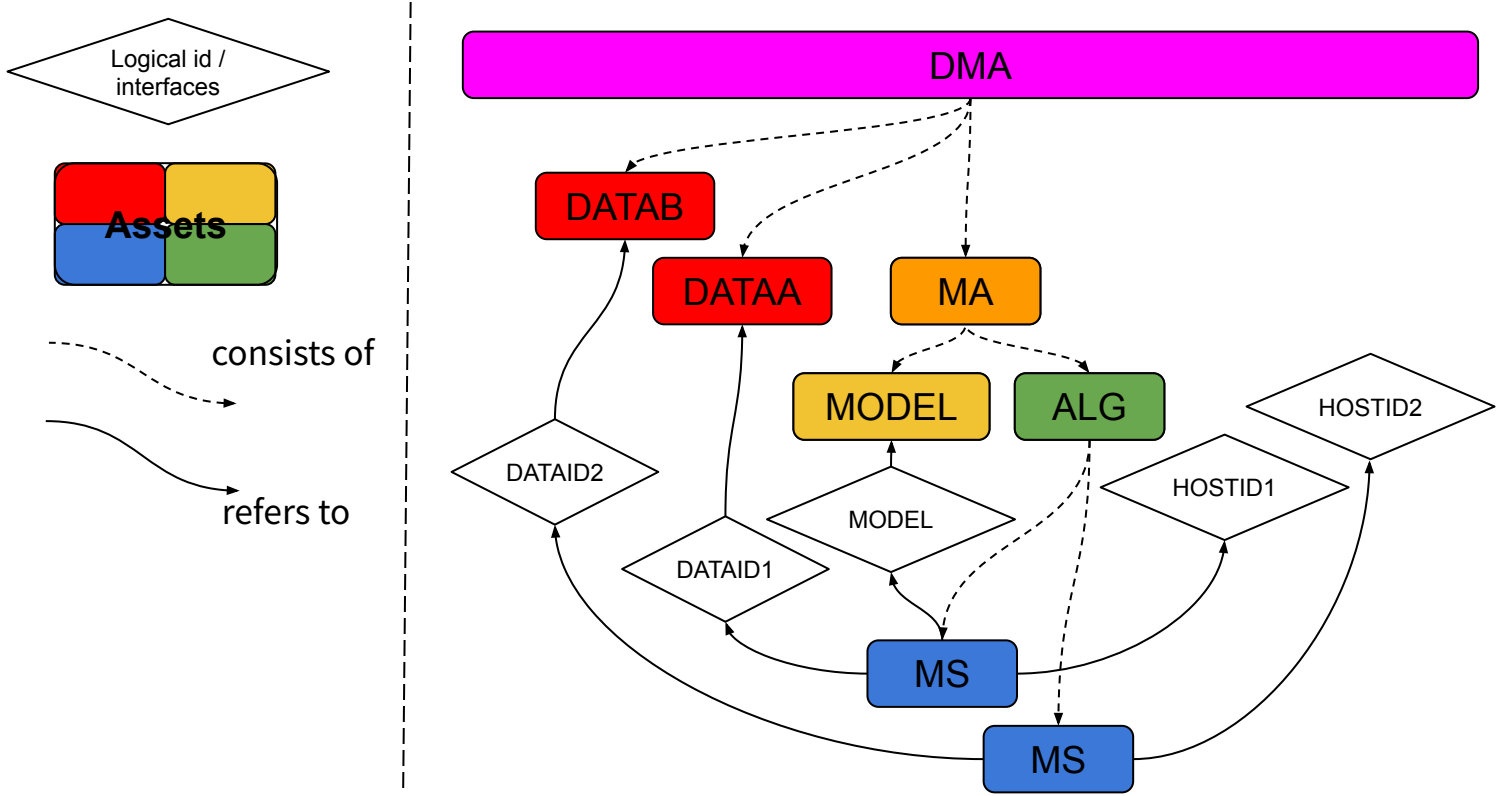
DMA Tuple Metadata

Concept	Subkey	Type	Example Value	Comment	Condition	Values
		Dates	R90/2021-05-01T00:00:00Z/PT48H	Days and hours the DMA Tuple will be active (optional)	optional	
		String	{'injectionMold': 'Circuit Case'}	User-defined key-value pairs: JSON string with additional information (optional)	optional	
Data Assets Mapping						
		[DataAssetsMapping](../dataassetsmapping.md)		Mapping required Data assets to Microservices specified in the MA Pair. Not every Microservice needs a Data asset.	mandatory	{"MSID_FLOWER": { "DATA1": 'DATAID_FLOWER"}}

DIGITBRAIN

DMA Tuple Metadata

Concept	Key	Subkey	Type	Example Value	Comment	Condition	Values
	Scope		Text	Effectiveness of the mold closing process	Short description of the scope of the DMA Tuple (human readable)	mandatory	"Structural Simulation for Engine Mount of Fraunhofer's
	IP Instance		ID / URI	ip_instance_123e4567-e89b-12d3	Identifier of the IP Instance the DMA Tuple is valid for (NB: Entity for IP Instance is referenced)	mandatory	"ID_IP_Instance"
	Namespace		ID / URI?	namespace_123e4567-e89b-12d3	Context to interpret the associated information	optional	
	MA Pair		ID / URI	"MAID_MYMA"	Identifier of the MA Pair associated to the DMA Tuple	mandatory	"MAID_RISTRA"
	Schedule		Dates	R90/2021-05-01T00:00:00Z/PT48H	Days and hours the DMA Tuple will be active	optional	
	Payload		String	{'injectionMold': 'Circuit Case'}	User-defined key-value pairs: JSON string with additional information (optional)	optional	
Data Assets Mapping							
	DataAssetsMapping		[DataAssetsMapping](../dataassetsmapping.md)	{ "MSID_MYMS_A": { "MY_SINK": "DATAID_MYDATA_A", "MY_STREAM": "DATAID_MYDATA_B" }	Mapping required Data assets to Microservices specified in the MA Pair. Not every Microservice needs a Data asset.	mandatory	{"MSID_ES_GUI": { "DATA1": "DATAID_RISTRA"}, "MSID_ES_MODELMANAGER": { "DATA2": "DATAID_RISTRA"}}


Logical id / interfaces
Assets
consists of
refers to
DMA
DATAB
DATAA
MA
MODEL
ALG
HOSTID2
DATAID2
HOSTID1
MODEL
DATAID1
MS
MS

Step 11: create host(s) in DMA

- In the DMA sheet, the value for “Deployments” is used to specify host(s) which represent virtual machine(s) that will execute your docker container(s)
- We have to **use the IDs defined in Algorithm** asset and define the parameters as shown on the right side.
- The only cloud provider that is supported in the metadata is the **cloudbroker**.
- The UUIDs (under “cloudbroker” section) **need to be taken from cloudbroker cloud environment**. Example values are as follows:

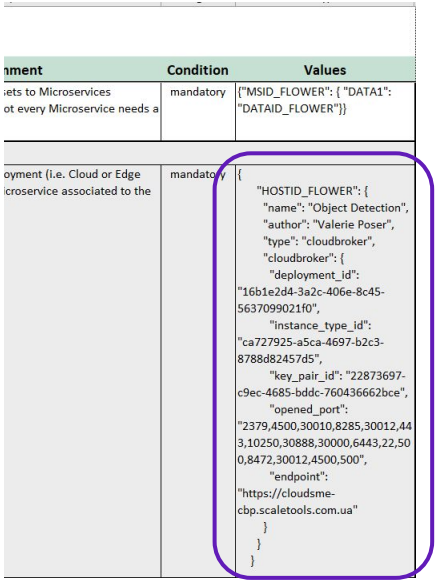
```
"deployment_id": "16b1e2d4-3a2c-406e-8c45-5637099021f0"
"instance_type_id": "ca727925-a5ca-4697-b2c3-8788d82457d5"
"key_pair_id": "22873697-c9ec-4685-bddc-760436662bce"
```

```
{
  "ID_OF_YOUR_HOST": {
    "name": "ANYNAME",
    "author": "ANYNAME",
    "type": "cloudbroker",
    "cloudbroker": {
      "deployment_id": "UUID",
      "instance_type_id": "UUID",
      "key_pair_id": "UUID",
      "opened_port": "22, 443, 80",
      "endpoint": "https://cloudsme-cbp.scaletools.com.ua"
    }
  }
}
```

- The port numbers in this example is just for illustration purpose. You need to define appropriate ports based on the requirements of your application.

Step 11: create host(s) in DMA (2)

- If you have more than one Microservice and would like to execute them on separate virtual machines, you need to create multiple hosts. You can do it as shown here.
- As mentioned, the UUIDs need to be taken from the cloudbroker (See Create a deployment in DMA for microservice(s)).



nment	Condition	Values
ets to Microservices ot every Microservice needs a	mandatory	{"MSID_FLOWER": { "DATA1": "DATAID_FLOWER"}}
oyment (i.e. Cloud or Edge icroservice associated to the	mandatory	{ "HOSTID_FLOWER": { "name": "Object Detection", "author": "Valerie Poser", "type": "cloudbroker", "cloudbroker": { "deployment_id": "16b1e2d4-3a2c-406e-8c45-5637099021f0", "instance_type_id": "ca727925-a5ca-4697-b2c3-8788d82457d5", "key_pair_id": "22873697-c9ec-4685-bddc-760436662bce", "opened_port": "2379,4500,30010,8285,30012,443,10250,30888,30000,6443,22,500,8472,30012,4500,500", "endpoint": "https://cloudsme-cbp.scaletools.com.ua" } } }

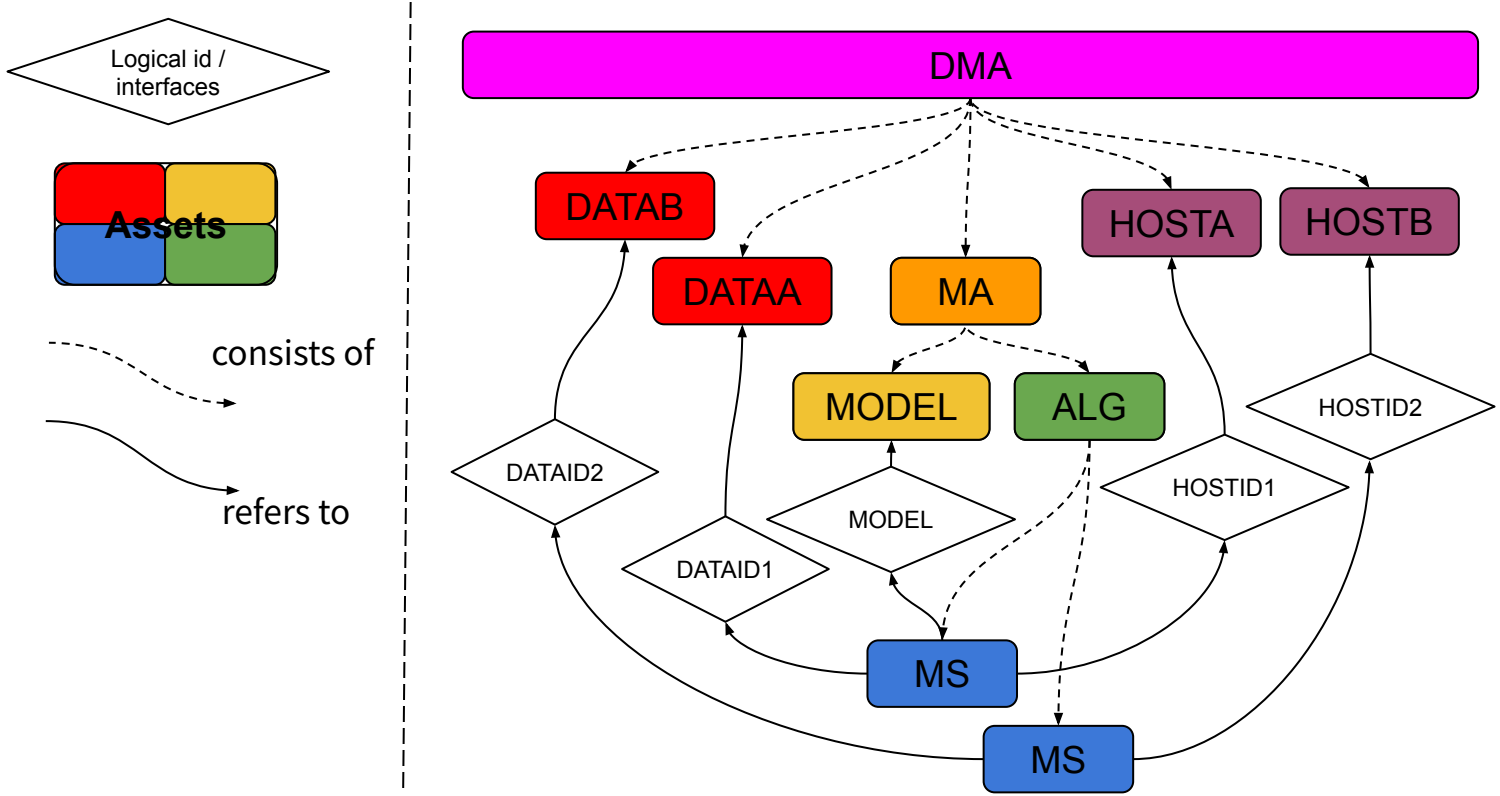
Check your Algorithm definition!
Assuming you specified two HOSTs in the **deploymentMapping of the Algorithm** asset such as:

```
{
    "MSID_ONE": "ID_OF_YOUR_HOST",
    "MSID_TWO": "ID_OF_ANOTHER_HOST"
}
```

Then the structure of the deployment should look like as on the right side (but with real values)

```
{
    "ID_OF_YOUR_HOST": {
      "name": "ANYNAME",
      "author": "ANYNAME",
      "type": "cloudbroker",
      "cloudbroker": {
        "deployment_id": "UUID",
        "instance_type_id": "UUID",
        "key_pair_id": "UUID",
        "opened_port": "22, 443, 80",
        "endpoint": "https://cloudsme-cbp.scaletools.com.ua"
      }
    },
    "ID_OF_ANOTHER_HOST": {
        "name": "ANYNAME",
        "author": "ANYNAME",
        "type": "cloudbroker",
        "cloudbroker": {
          "deployment_id": "UUID",
          "instance_type_id": "UUID",
          "key_pair_id": "UUID",
          "opened_port": "22, 443, 80",
          "endpoint": "https://cloudsme-cbp.scaletools.com.ua"

  }
}
```


Logical id / interfaces
Assets
consists of
refers to
DMA
DATAB
DATAA
MA
HOSTA
HOSTB
MODEL
ALG
DATAID2
DATAID1
MODEL
HOSTID1
HOSTID2
MS
MS

Conclusion

- This presentation introduced a **method to create DMA tuples manually by using an excel table** to store and organise the required key values
- This is a **temporary solution**, in a later stage of the project, the Publishing Interface (PI) will guide you through the creation and linking of the different assets.
- By creating the excel sheet representing your DMA
  - you **help the developers** to test the tools responsible for generating the application
  - you help yourself to **understand the concept** of DIGITbrain
  - you help yourself to **create your DMA** that should be uploaded to PI later

- The slides presented here **focused on the key values needed to generate a runnable** application/service
- Do not forget to fill in **other relevant parameters** in each asset
- To understand the usage/meaning of the other parameters, please use the DIGITbrain documentation website at

  https://github.com/DIGITbrain/digitbrain.github.io