Workflow run RO-crate draft

Notes/links

- Previous work:
  - Workflow Run requirements
  - Workflow definition in RO-Crate:
    - https://www.researchobject.org/ro-crate/1.1/workflows.html
    - https://about.workflowhub.eu/Workflow-RO-Crate/
    - https://bioschemas.org/profiles/ComputationalWorkflow/
  - Provenance of software run in RO-Crate
    - https://www.researchobject.org/ro-crate/1.1/provenance.html
  - Workflow Testing RO-Crate
    - https://github.com/crs4/life_monitor/wiki/Workflow-Testing-RO-Crate
  - CWLProv paper have many general recommendations on provenance
    - from Table 1:

1. Save and share **all parameters** used for **each software** executed in a given workflow (including **default values** of parameters used) [55, 57–59].
2. Avoid manual processing of data, and if using shims [60], then make these part of the workflow to **fully automate** the computational process [57, 59].
3. Include **intermediate results** where possible when publishing an analysis [55, 58, 59].
4. Record the exact **software versions** used [57, 59].
5. If using public data (**reference data,** variant databases), then it is necessary to store and share the actual **data versions** used [3, 61, 57, 59].
6. Workflows should be **well-described**, **annotated**, and offer associated **metadata**. Annotations such as user-contributed tags and **versions** should be assigned to workflows and shared when publishing the workflows and associated results [8, 12, 55, 62, 63].
7. Use and store **stable identifiers** for all artefacts including the **workflow**, the **datasets**, and the **software components** [62, 63].
8. Share the details of the **computational environment** [8, 61,
9. Share **workflow specifications/descriptions** used in the analysis [8, 55, 58, 63, 64].
10. Aggregate the **software** with the analysis and share this when publishing a given analysis [61, 63, 64].
11. Share **raw data** used in the analysis [8, 55, 58, 63, 64].
12. Store all **attributions** related to data resources and software systems used [55,64].
13. **Workflows** should be preserved along with the **provenance trace** of the data and results [8, 12, 55, 59, 64].
14. **Data flow diagrams** of the computational analysis using workflows should be provided [58, 61].
15. **Open source licensing** for methods, software, code, workflows, and data should be adopted instead of proprietary resources [58, 59, 63, 64, 66].
16. **Data**, **code**, and all **workflow steps** should be shared in a format that others can easily understand, preferably in a **system-neutral** language [8, 58, 66].
17. Promote **easy execution of workflows** without making significant changes to the underlying environment [3].
18. Information about **compute** and **storage resources** should be stored and shared as part of the workflow [61].
19. **Example input** and **sample output** data should be preserved and published along with the workflow-based analysis [8, 65].

- Two needs:
    - workflow executions to be executed (prospective)
    - workflow executions that have been executed (retrospective)
- Start simple:
    - One workflow is sent to one engine for running and it worked.
    - Then gradually we can add from Workflow Run requirements, e.g.:
        - Where was it executed?
        - When was it executed?
        - What wf engine capabilities are needed (e.g. Docker)
        - Version of workflow engine
        - Input / outputs as a mixed bag of files
        - Input / outputs linked to corresponding FormalParameter definitions
        - Errors and logs
        - Steps executed
        - Container images used
        - Command lines executed
    - What are the minimum requirements for to be executed jobs?
        - Should correspond to GA4GH WES API fields?
    - Open questions:
        - Identifying inputs/outputs with TRS API?
        - Workflow definition embedded or as nested RO-Crate? (e.g. reference to workflowhub.eu RO-Crate)
    - Possibly related tickets on GA4GH:
        - https://github.com/ga4gh/workflow-execution-service-schemas/issues/163
        - https://github.com/ga4gh/cloud-interop-testing/issues/111
        - → Keep GA4GH call in the loop of what we're doing

Draft JSON-LD sections

Go "free-hand" at first, then we see if we can align against schema.org, bioschemas or GA4GH API.

…

Note: After example 1 including boiler plate, only show the JSON blocks to add to `"@graph": []`

Example 1: One workflow is sent to one engine for running and it worked

```
{ "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [
    {
      "@type": "CreativeWork",
      "@id": "ro-crate-metadata.json",
      "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
      "about": {"@id": "./"}
    },
    {
      "@id": "./",
```

```
      "@type": "Dataset",
      "hasPart": [
          { "@id": "workflow/retropath.knime" }
      ]
    },
    { "@id": “#workflow-run”,
      "@type": “?”
    }


]}
```

Example 2: Where was it executed, when?

```
{ "@id": “#workflow-run”,
  "@type": “?”
}
```

Example 3: What workflow engine capabilities are needed/used

(e.g. Docker)

```
{ "@id": “#engine”,
  "@type": “?”:
}
```

Example 4: Name and version of workflow engine

Example 5: Just a bunch of inputs/outputs

Example 6: Inputs/outputs linked to FormalParameter definitions

Example 7: Errors and logs

Example 8: Steps executed

Example 9: Container images / packages used

Example 10: Command line details

Existing approaches

ComputationalWorkflow profile according to RO-Crate 1.1 spec

From
https://www.researchobject.org/ro-crate/1.1/workflows.html#complying-with-bioschemas-computational-workflow-profile

```
{ "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [
    {
      "@type": "CreativeWork",
      "@id": "ro-crate-metadata.json",
      "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
      "about": {"@id": "./"}
    },
    {
      "@id": "./",
      "@type": "Dataset",
      "hasPart": [
          { "@id": "workflow/retropath.knime" }
      ]
    },
    {
      "@id": "workflow/alignment.knime",
      "@type": ["File", "SoftwareSourceCode", "ComputationalWorkflow"],
      "conformsTo":
        {"@id":
"https://bioschemas.org/profiles/ComputationalWorkflow/0.5-DRAFT-2020_07_21/"},
      "name": "Sequence alignment workflow",
      "programmingLanguage": {"@id": "#knime"},
      "creator": {"@id": "#alice"},
      "dateCreated": "2020-05-23",
      "license": { "@id": "https://spdx.org/licenses/CC-BY-NC-SA-4.0"},
      "input": [
        { "@id": "#36aadbd4-4a2d-4e33-83b4-0cbf6a6a8c5b"}
      ],
      "output": [
        { "@id": "#6c703fee-6af7-4fdb-a57d-9e8bc4486044"},
        { "@id": "#2f32b861-e43c-401f-8c42-04fd84273bdf"}
      ],
      "sdPublisher": {"@id": "#workflow-hub"},
      "url": "http://example.com/workflows/alignment",
```

```
      "version": "0.5.0"
    },
    {
      "@id": "#36aadbd4-4a2d-4e33-83b4-0cbf6a6a8c5b",
      "@type": "FormalParameter",
      "conformsTo": {"@id":
"https://bioschemas.org/profiles/FormalParameter/0.1-DRAFT-2020_07_21/"},
      "name": "genome_sequence",
      "valueRequired": true,
      "additionalType": {"@id": "http://edamontology.org/data_2977"},
      "format": {"@id": "http://edamontology.org/format_1929"}
    },
    {
      "@id": "#6c703fee-6af7-4fdb-a57d-9e8bc4486044",
      "@type": "FormalParameter",
      "conformsTo": {"@id":
"https://bioschemas.org/profiles/FormalParameter/0.1-DRAFT-2020_07_21/"},
      "name": "cleaned_sequence",
      "additionalType": {"@id": "http://edamontology.org/data_2977"},
      "encodingFormat": {"@id": "http://edamontology.org/format_2572"}
    },
    {
      "@id": "#2f32b861-e43c-401f-8c42-04fd84273bdf",
      "@type": "FormalParameter",
      "conformsTo": {"@id":
"https://bioschemas.org/profiles/FormalParameter/0.1-DRAFT-2020_07_21/"},
      "name": "sequence_alignment",
      "additionalType": {"@id": "http://edamontology.org/data_1383"},
      "encodingFormat": {"@id": "http://edamontology.org/format_1982"}
    },
    {
      "@id": "https://spdx.org/licenses/CC-BY-NC-SA-4.0",
      "@type": "CreativeWork",
      "name": "Creative Commons Attribution Non Commercial Share Alike 4.0
International",
      "alternateName": "CC-BY-NC-SA-4.0"
    },
    {
      "@id": "#knime",
      "@type": "ProgrammingLanguage",
      "name": "KNIME Analytics Platform",
      "alternateName": "KNIME",
      "url": "https://www.knime.com/whats-new-in-knime-41",
      "version": "4.1.3"
    },
    {
      "@id": "#alice",
      "@type": "Person",
      "name": "Alice Brown"
    },
    {
```

```
      "@id": "#workflow-hub",
      "@type": "Organization",
      "name": "Example Workflow Hub",
      "url":"http://example.com/workflows/"
    },
    {
      "@id": "http://edamontology.org/format_1929",
      "@type": "Thing",
      "name": "FASTA sequence format"
    },
    {
      "@id": "http://edamontology.org/format_1982",
      "@type": "Thing",
      "name": "ClustalW alignment format"
    },
    {
      "@id": "http://edamontology.org/format_2572",
      "@type": "Thing",
      "name": "BAM format"
    },
    {
      "@id": "http://edamontology.org/data_2977",
      "@type": "Thing",
      "name": "Nucleic acid sequence"
    },
    {
      "@id": "http://edamontology.org/data_1383",
      "@type": "Thing",
      "name": "Nucleic acid sequence alignment"
    }
  ]
}
```

Retrospective Workflow Run RO-Crates

Software used to create - from RO-Crate 1.1

From https://www.researchobject.org/ro-crate/1.1/provenance.html#software-used-to-create-files

*In the below example, an image with the @id of pics/2017-06-11%2012.56.14.jpg was transformed into an **new image pics/sepia_fence.jpg** using the ImageMagick software application as "instrument". Actions MAY have human-readable names, which MAY be machine generated for use at scale.*

```
    {
      "@id": "#SepiaConversion_1",
      "@type": "CreateAction[1]",
```

[1] There are different subclasses of Action that may be relevant to provenance.

```
      "name": "Convert dog image to sepia",
      "description": "convert -sepia-tone 80% test_data/sample/pics/2017-06-11\\
12.56.14.jpg test_data/sample/pics/sepia_fence.jpg",
      "endTime": "2018-09-19T17:01:07+10:00",
      "instrument": {
        "@id": "https://www.imagemagick.org/"
      },
      "object": {
        "@id": "pics/2017-06-11%2012.56.14.jpg"
      },
      "result": {
        "@id": "pics/sepia_fence.jpg"
      }
    },
{
      "@id": "https://www.imagemagick.org/",
      "@type": "SoftwareApplication",
      "url": "https://www.imagemagick.org/",
      "name": "ImageMagick",
      "version": "ImageMagick 6.9.7-4 Q16 x86_64 20170114
http://www.imagemagick.org"
}
```


CWLProv

https://github.com/common-workflow-language/cwlprov/ predates RO-Crate and use of schema.org, and uses only W3C PROV https://www.w3.org/TR/prov-overview/ with a lighter Research Object manifest.

We can re-use from this approach:

- Hash-based identifiers
    - Content-addressable files (although people like to keep filenames!)
- arcp identifiers to identify any files that are not directly URI accessible
    - .. or will TRS API solve this?

See also CWLProv paper which have many more general recommendations on provenance.

In manifest.json


```
{
            "uri": "urn:uuid:9ed6545d-dfb2-4cab-b4af-102735a2661e",
            "about": "urn:uuid:1f767ad4-ac52-4623-b5bc-dd9faf2b869f",
```

```
            "content": [
                "provenance/primary.cwlprov.xml",
                "provenance/primary.cwlprov.json",
                "provenance/primary.cwlprov.provn",
                "provenance/primary.cwlprov.ttl",
                "provenance/primary.cwlprov.jsonld",
                "provenance/primary.cwlprov.nt"
            ],
            "oa:motivatedBy": {
                "@id": "http://www.w3.org/ns/prov#has_provenance"
            }
        }
1f767ad4-ac52-4623-b5bc-dd9faf2b869f is here the workflow run ID
       {
            "uri": "provenance/primary.cwlprov.jsonld",
            "mediatype": "application/ld+json",
            "conformsTo": [
                "http://www.w3.org/TR/2013/REC-prov-o-20130430/",
                "https://w3id.org/cwl/prov/0.6.0"
            ],
            "createdOn": "2018-10-25T15:46:43.192069",
            "createdBy": {
                "uri": "urn:uuid:ac9c1653-4291-47bc-86f8-6dedcff13519",
                "name": "cwltool 1.0.20181012180214"
            }
        }
```

Ac9c1653-4291-47bc-86f8-6dedcff13519 is here the run of the workflow engine (which is different from its run of the workflow!)

Each data file has a content-adressable identifier and is (hopefully) also included under data/

```
"aggregates": [
```

```
        {
            "uri": "urn:hash::sha1:327fc7aedf4f6b69a42a7c8b808dc5a7aff61376",
            "bundledAs": {
                "uri":
"arcp://uuid,1f767ad4-ac52-4623-b5bc-dd9faf2b869f/data/32/327fc7aedf4f6b69a42a7c8b8
08dc5a7aff61376",
                "folder": "/data/32/",
                "filename": "327fc7aedf4f6b69a42a7c8b808dc5a7aff61376"
            }
        },
        {
            "uri": "urn:hash::sha1:97fe1b50b4582cebc7d853796ebd62e3e163aa3f",
            "bundledAs": {
                "uri":
"arcp://uuid,1f767ad4-ac52-4623-b5bc-dd9faf2b869f/data/97/97fe1b50b4582cebc7d853796
ebd62e3e163aa3f",
                "folder": "/data/97/",
                "filename": "97fe1b50b4582cebc7d853796ebd62e3e163aa3f"
            }
        },
```


metadata/provenance contains one PROV file per (nested) workflow run (saved in multiple formats), which refer to how these data identifiers `urn:hash::sha1:327fc7aedf4f6b69a42a7c8b808dc5a7aff61376` were created by steps in workflow run `urn:uuid:1f767ad4-ac52-4623-b5bc-dd9faf2b869f`.

Below I show some of the metadata/provenance/primary.cwlprov.jsonld - note that this JSON-LD has no @context and therefore is very verbose.


```
 {
    "@id": "urn:hash::sha1:327fc7aedf4f6b69a42a7c8b808dc5a7aff61376",
    "@type": [
      "http://purl.org/wf4ever/wfprov#Artifact",
      "http://www.w3.org/ns/prov#Entity"
    ]
  }
{
       "@id": "urn:uuid:6e84364f-faa9-4a27-aaba-5e4b80d9564b",
       "@type": [
       "http://purl.org/wf4ever/wfprov#Artifact",
       "http://purl.org/wf4ever/wf4ever#File",
       "http://www.w3.org/ns/prov#Entity"
       ],
       "http://www.w3.org/ns/prov#specializationOf": [
       {
       "@id": "urn:hash::sha1:327fc7aedf4f6b69a42a7c8b808dc5a7aff61376"
       }
       ],
       "https://w3id.org/cwl/prov#basename": [
       {
```

```
      "@value": "whale.txt"
      }
      ],
      "https://w3id.org/cwl/prov#nameext": [
      {
      "@value": ".txt"
      }
      ],
      "https://w3id.org/cwl/prov#nameroot": [
      {
      "@value": "whale"
      }
      ]
  }
  {
      "@id": "urn:uuid:fe16801a-7995-4968-a8bb-5e9d46255bb7",
      "@type": [
      "http://www.w3.org/ns/prov#Entity",
      "http://purl.org/wf4ever/wfprov#Artifact",
      "http://purl.org/wf4ever/wf4ever#File"
      ],
      "http://www.w3.org/ns/prov#specializationOf": [
      {
      "@id": "urn:hash::sha1:327fc7aedf4f6b69a42a7c8b808dc5a7aff61376"
      }
      ],
      "https://w3id.org/cwl/prov#basename": [
      {
      "@value": "whale.txt"
      }
      ],
      "https://w3id.org/cwl/prov#nameext": [
      {
      "@value": ".txt"
      }
      ],
      "https://w3id.org/cwl/prov#nameroot": [
      {
      "@value": "whale"
      }
      ]
  }
```


Why are there two **specializations** with their own UUIDs? They are kind of two occurrences of the same bytes within the workflow, but potentially with different filenames and bindings at different ports. Keeping these separate is kind of more important for **generation**, because then each entity is only generated once, rather than at two different points in time (when the workflow finishes vs when a step finishes), which gives inconsistent provenance.

Here is the usage by the overall workflow input (role):


```
  {
```

```
      "@id": "_:Nee6d99bc856a4ddf8c7a7ab6afcfc7ef",
      "@type": [
      "http://www.w3.org/ns/prov#Usage"
      ],
      "http://www.w3.org/ns/prov#atTime": [
      {
      "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
      "@value": "2018-10-25T15:46:35.303484"
      }
      ],
      "http://www.w3.org/ns/prov#entity": [
      {
      "@id": "urn:uuid:fe16801a-7995-4968-a8bb-5e9d46255bb7"
      }
      ],
      "http://www.w3.org/ns/prov#hadRole": [
      {
      "@id":
"arcp://uuid,1f767ad4-ac52-4623-b5bc-dd9faf2b869f/workflow/packed.cwl#main/input"
      }
      ]
  }
```


And here is the usage at the step rev as input ‘input’


```
  {
      "@id": "_:N34d7ba32bf1f4353a4fcb936dfe86773",
      "@type": [
      "http://www.w3.org/ns/prov#Usage"
      ],
      "http://www.w3.org/ns/prov#atTime": [
      {
      "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
      "@value": "2018-10-25T15:46:35.597726"
      }
      ],
      "http://www.w3.org/ns/prov#entity": [
      {
      "@id": "urn:uuid:6e84364f-faa9-4a27-aaba-5e4b80d9564b"
      }
      ],
      "http://www.w3.org/ns/prov#hadRole": [
      {
      "@id":
"arcp://uuid,1f767ad4-ac52-4623-b5bc-dd9faf2b869f/workflow/packed.cwl#main/rev/inpu
t"
      }
      ]
  }
```

Such distinction between ‘inside’ and ‘outside’ may be needed in Workflow Run RO-Crate when dealing with services like WES, because there is one set of provenance at WES level and another at workflow level, and the input names and filenames may not be the same.