Workflow definition template

1. Application

MADF	Pilot	Test	Benchmark

2. Requirements

ID	Type	Location	Description

3. Outputs

ID	Type	Location	Description

4. Steps

Step ID	Command	Description

5. Workflow

Step ID	Depends on	Required by	Success definition	Conditional jump

6. Resources

Step ID	Cores	Nodes	GPU	RAM	Time	# runs

7. Appendix

Example

The goal of this template is to obtain reproducible workflows for all MADFs, pilots, tests and benchmarks involved in the MSO4SC project.

These workflows will be implemented in the future by TOSCA files and executed by the orchestrator.

Note: *From the orchestrator point of view, the more steps an application can be splitted into, the more control the orchestrator have over it. This means that an application execution, seen as a workflow of "atomic" executions, can be optimised by the orchestrator taking benefit of automatic decisions "on the fly" such as the execution of the jobs in the best infrastructure available, more efficient data allocation, or error management of the differents steps.*

As an example, a monolithic application with only one step will need to ask to the HPC for the maximum resources it needs during its execution, taking long to start. Also if it has any error during it, the execution will end and the user will have to reconfigure and run it again from the start. On the other hand, if the same application is splitted into smaller tasks, some of them sequential, some of them parallel, the orchestrator will ask for the precise resources each one need, and will put them in the more suitable HPC and partition to run. Also if any of these tasks fails, the orchestrator can rerun the step following the rules defined previously by the user in the orchestrator (e.g, reconfiguring something).

If you need to add any extra information or attachment to the workflow, please use the **Appendix** section at the end of the template.

Your help is welcome! Please add any comment or suggestion in order to improve this document.

Workflow definition

In this example an hypothetical test application is commented.

1. Application

This section is used to unambiguously identify the target application of each workflow. The target application can be a MADF, a pilot, a test or a benchmark.

Note:

- *"Test" and "Benchmark" columns are exclusive.*

MADF	Pilot	Test	Benchmark
Feel++	HiFiMagnet	HDG solver (?)	-

2. Requirements

This section defines the environment in which the workflow will be performed.

This section contains all the components involved in the workflow. This is a heterogeneous table where to define and locate which sources, requirements, data or processes are necessary in order to execute the complete workflow.

Some examples of the heterogeneous components that can be taken into account in this section are shown below.

Note: *in order to ease reproducibility and portability, It's strongly recommended to include all the necessary components within a container when possible.*

Req ID	Type	Location	Description
REQ-0	Sources	*http://blahblah*/myapp.tar.gz	Mesh generator source code
REQ-1	Compilation process	*FT2 [REQ-0]*	**Command:** *./compile_myapp.sh* (*1 Appendix)
REQ-2	Execution	*FT2 [REQ-1]*	Mesh generation **Command:** ./myapp -o *REQ-3*
REQ-3	I/O data	(customizable)	Mesh file (output from *REQ-2*)
REQ-4	Container	**Dockerhub:** *docker:*//blahblah **Path:** *FT2:*//DIR/gromacs.img	My container with some apps
REQ-5	Input data	*Container [REQ-4]*://opt/test_data	Some input data

3. Steps

This section defines the atomic steps included in the workflow. With this table we want to know who is involved in every action and how this action is performed.

Note:

- Steps could be related with previously defined ***Requirements*** (Section 2)
- Steps can be a simple command or complex scripts that can be attached in the ***Appendix*** section.

Step ID	Command
STEP-0	**FT2:** module load blahblah; /opt/cesga/1st_step *REQ-3*
STEP-1	**Container[***REQ-4***]**://usr/local/bin/2nd_step
STEP-2	**Container[***REQ-4***]**://usr/local/bin/3rd_step *REQ-5* $HOME/output_dir

4. Workflow

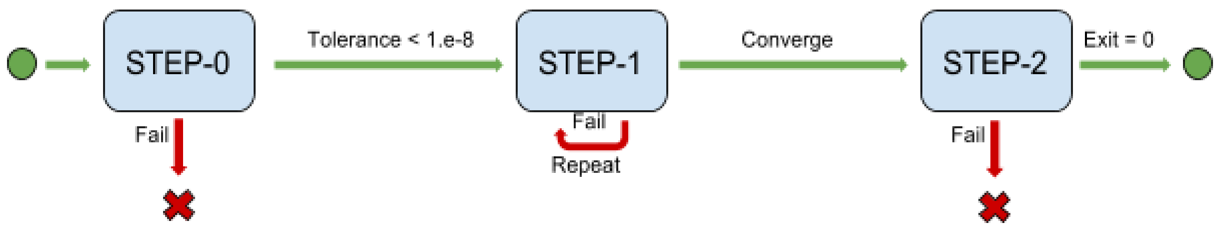
This table represents a directed graph defining the workflow. The order and relationship of the steps is defined by the "*Depends on*", "*Required by*" and "*Conditional jump*" columns.

"Success definition" column describes the condition to accept each step execution as successful. It can depend on an exit code, in a log message, etc.

Note: "*Depends on*", "*Required by*" and "*Conditional jump*" columns can contain several steps per cell.

Step ID	Depends on	Required by	Success definition	Conditional jump
STEP-0	-	STEP-1	**Log:** Tolerance < 1.e-8	No
STEP-1	STEP-0	STEP-2	**Log:** Convergence	STEP-1 (Repeat if not success)
STEP-2	STEP-1	-	**Exit code:** 0	No

The following graph represents the workflow defined in the table above:



5. Resources

Computational resources used by each step. This resources per step will be allocated by the resource manager (e.g. Slurm).

Step ID	Cores	Nodes	GPU	RAM	Time	# runs
STEP-0	1	1	0	256 MB	00:10:00	1
STEP-1	24	1	0	6 GB	01:00:00	61
STEP-2	6	1	0	1 GB	00:10:00	61

6. Appendix

Any extra information can be added in this section.

1. *compile_myapp.sh*:
```
cd  tmp
wget http://blahblah/myapp.tar.gz
tar xzf myapp.tar.gz
cd /tmp/myapp
make install
```

7. Open lines

- Will we need to represent checkpointing?
- Will we need to represent iterative processes over entire workflows?
- AOB

Fenics-HPC:

FEniCS-HPC is based on an adaptive mesh refinement methodology where the computational mesh is **iteratively refined** by solving the primal (usual) problem, and a dual problem providing sensitivity information for the mesh refinement. An ensemble-averaging framework is also being developed, requiring multiple realizations of the same primal simulation.

This means the orchestration has additional staging opportunities, which are described below, and the typical separation into pre-processing (with mesh generation) and processing (with computation of solutions on the mesh) do not hold, but are intermixed.

- Processing
    - Different iterations in the adaptive sequence can be run on different infrastructures, where later iterations in the adaptive sequence use more refined meshes and require more computational resources. However, iteration i+1 depends on iteration i;
    - An ensemble simulation can be split, where the different realizations of the same primal computation can be run on different infrastructures;
    - Checkpoint restart within time loops for time-dependent applications. Time interval computation can be scheduled.

Post-processing

- Compute new data sets from simulation data such as Q-criterion (vorticity measure);typically embarrassingly parallel across time samples.

Calendar/meetings:

Feel++: Have a meeting to fill the document

One folder with one document for each use-case:

- https://drive.google.com/drive/folders/0BxR2J3_0QqevNmVvZE9JTkxxMGM