Hewlett Packard Enterprise

HPC workflow orchestration using the ipython notebook platform

pres139

Jonathan Sparks, Technologist, Cloud Solutions
Ayad Jassim, Cloud Solutions

May 11, 2023

Motivation

- Simplify the orchestration of complex workflows
- Present a new onboarding experience, “download, develop, and deploy”
- Adaptable and flexible environment
- Standard-based platform
- Leverage off-the-shelf components
- Framework-as-a-Service
- Curated marketplace of scientific/data workflows

Vision

Environment

Easily consumed workflows

- Core components
  - A marketplace of workflows (workloads, applications, data)
  - Workflow templates
  - Developer support (Visual Studio, environments, debug, and deploy)

Frameworks

Orchestration of applications

- Core components
  - Application catalogs (pre-built, optimized for the platform)
  - Development environments (pre-built or customizable)
  - Choice of execution environments, and orchestrators

Bring your own

Download, develop, and deploy

- Core components
  - Templates
  - Framework-as-a-service
  - Workflow marketplace

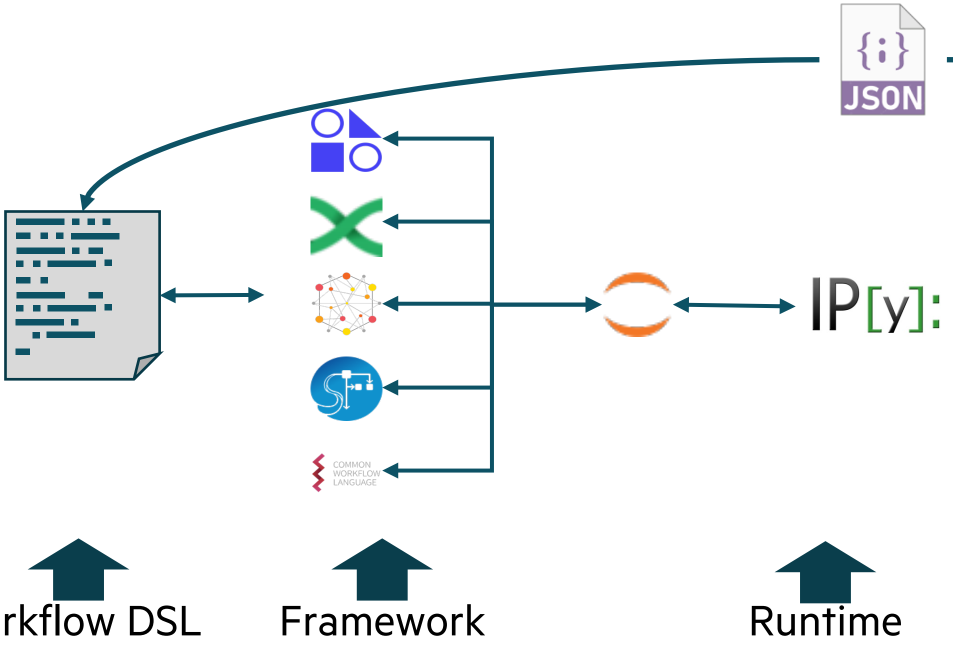
Workflow Requirements

- Language – job abstraction, good semantics
- Machine-independent (as much as possible), infrastructure agnostic – portable
- Flexible resources models – local, remote execution (edge/core/cloud)
- Multi-language support – polyglot (application-specific DSL/interfacing)
- Consumable – containers, registries, repositories

Workflow System

- **Workflow** Interface, description, DSL

- **Framework** provides the runtime, such as Nextflow, ExaWorks, Streamflow CWL, etc.

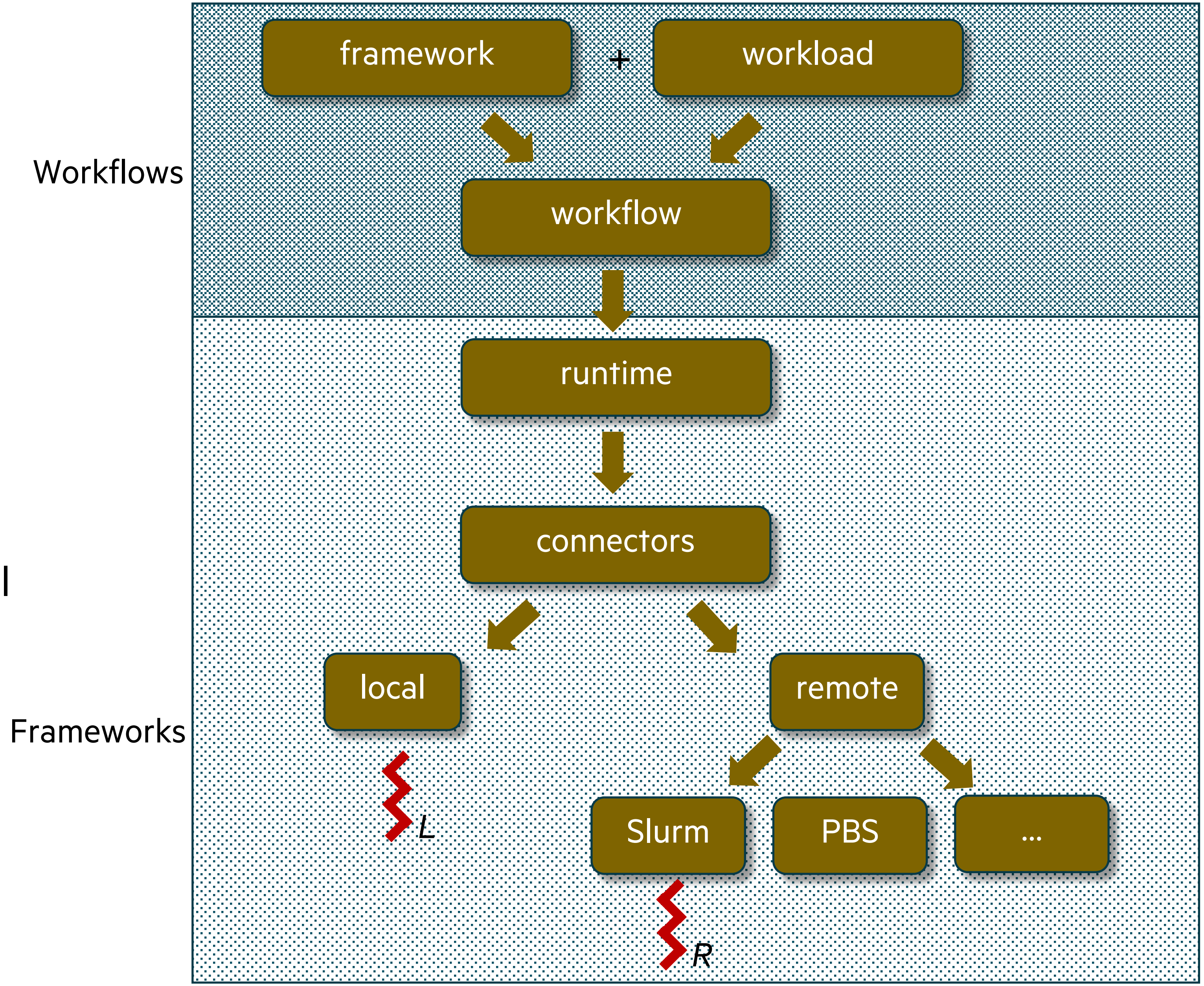


```
"org.hpe.staxs.cpu": "x86_64",
"org.hpe.staxs.framework": ”nextflow",
"org.hpe.staxs.gpu": "none",
"org.hpe.staxs.kabi": "2.28",
"org.hpe.staxs.mpi": ”mpich",
"org.hpe.staxs.network": "none",
"org.hpe.staxs.os": "Red Hat Enterprise Linux 8.7 (Ootpa)",
"org.hpe.staxs.platform": "jupyter",
```

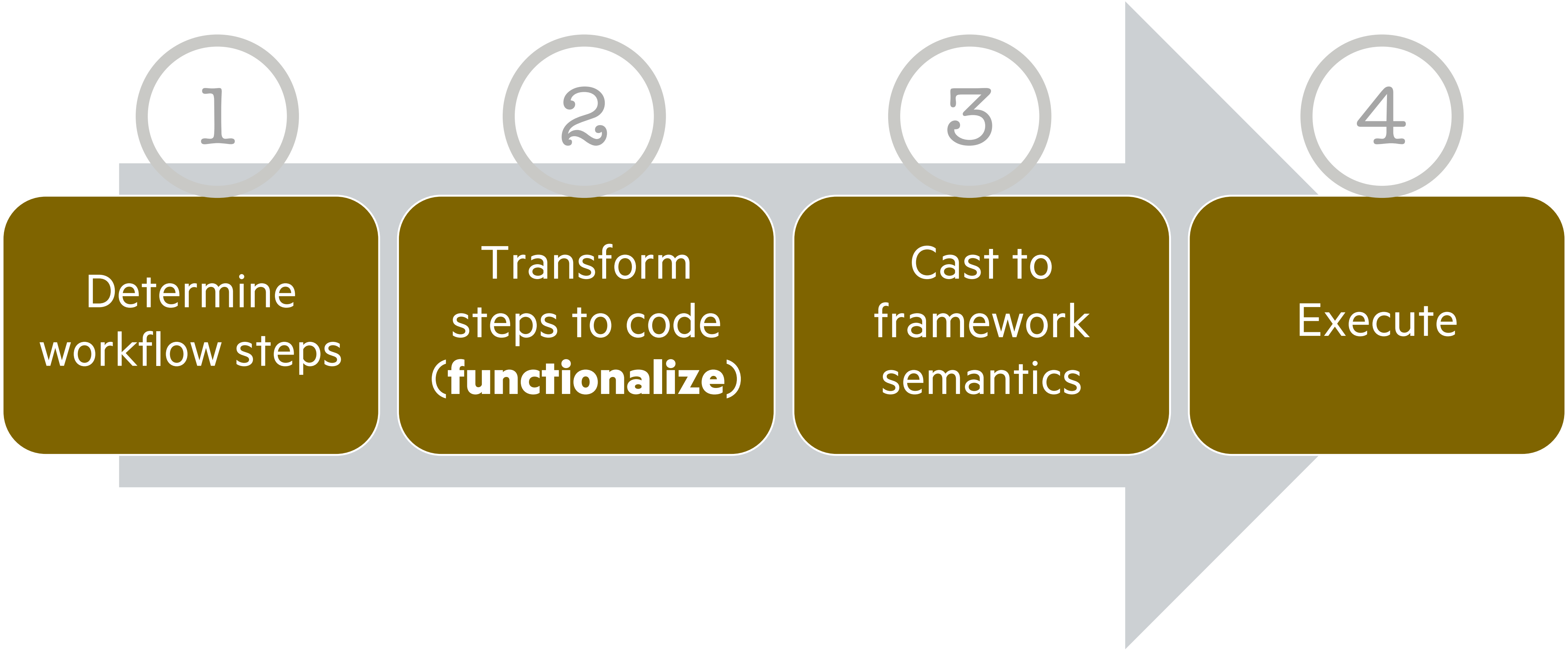
Workflows & Frameworks

Workflow

Framework provides the computational environment



Workflow Development



1 Determine workflow steps

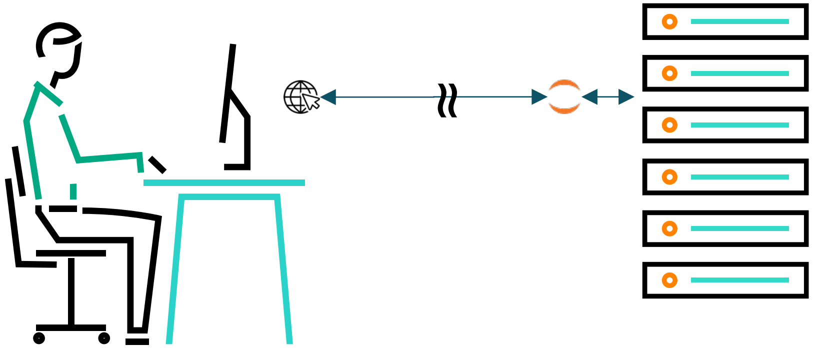
2 Transform steps to code (**functionalize**)

3 Cast to framework semantics

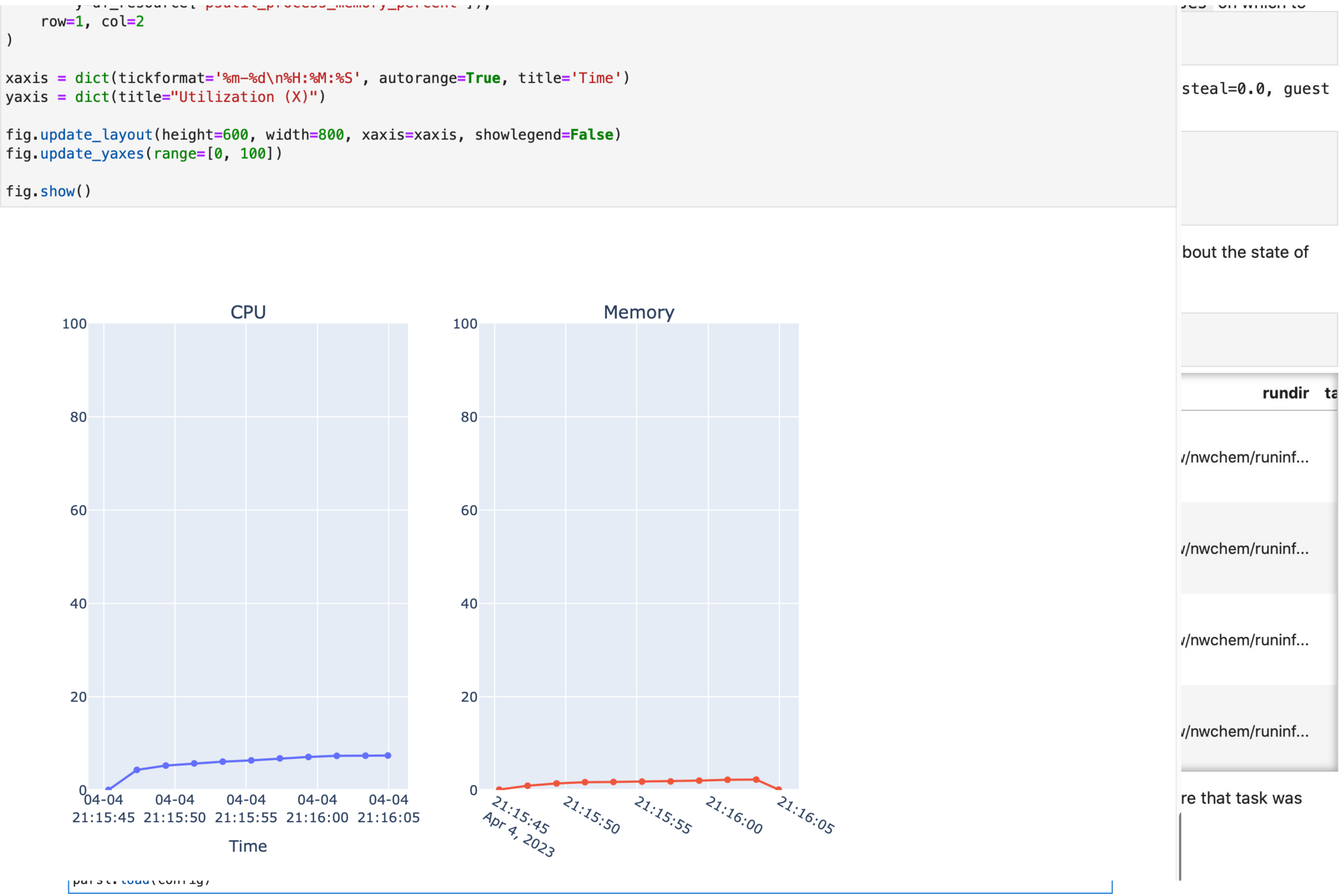
4 Execute

Use Case 1 – ExaWorks NWChem

- Framework: ExaWorks
- Application/workflow: nwChem



Use Case 1 – ExaWorks NWChem



```
    y=df_resource['psutil_process_memory_percent']),
    row=1, col=2
)

xaxis = dict(tickformat='%m-%d\n%H:%M:%S', autorange=True, title='Time')
yaxis = dict(title="Utilization (X)")

fig.update_layout(height=600, width=800, xaxis=xaxis, showlegend=False)
fig.update_yaxes(range=[0, 100])

fig.show()
```

```
parsl.load(config)
```

steal=0.0, guest

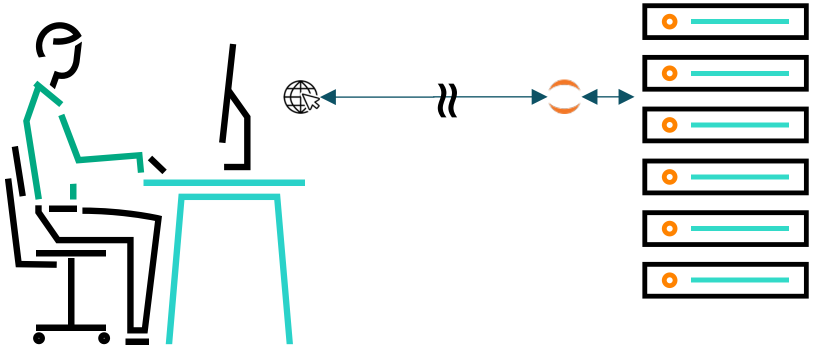
bout the state of

rundir	ta
/nwchem/runinf...	
/nwchem/runinf...	
/nwchem/runinf...	
/nwchem/runinf...	

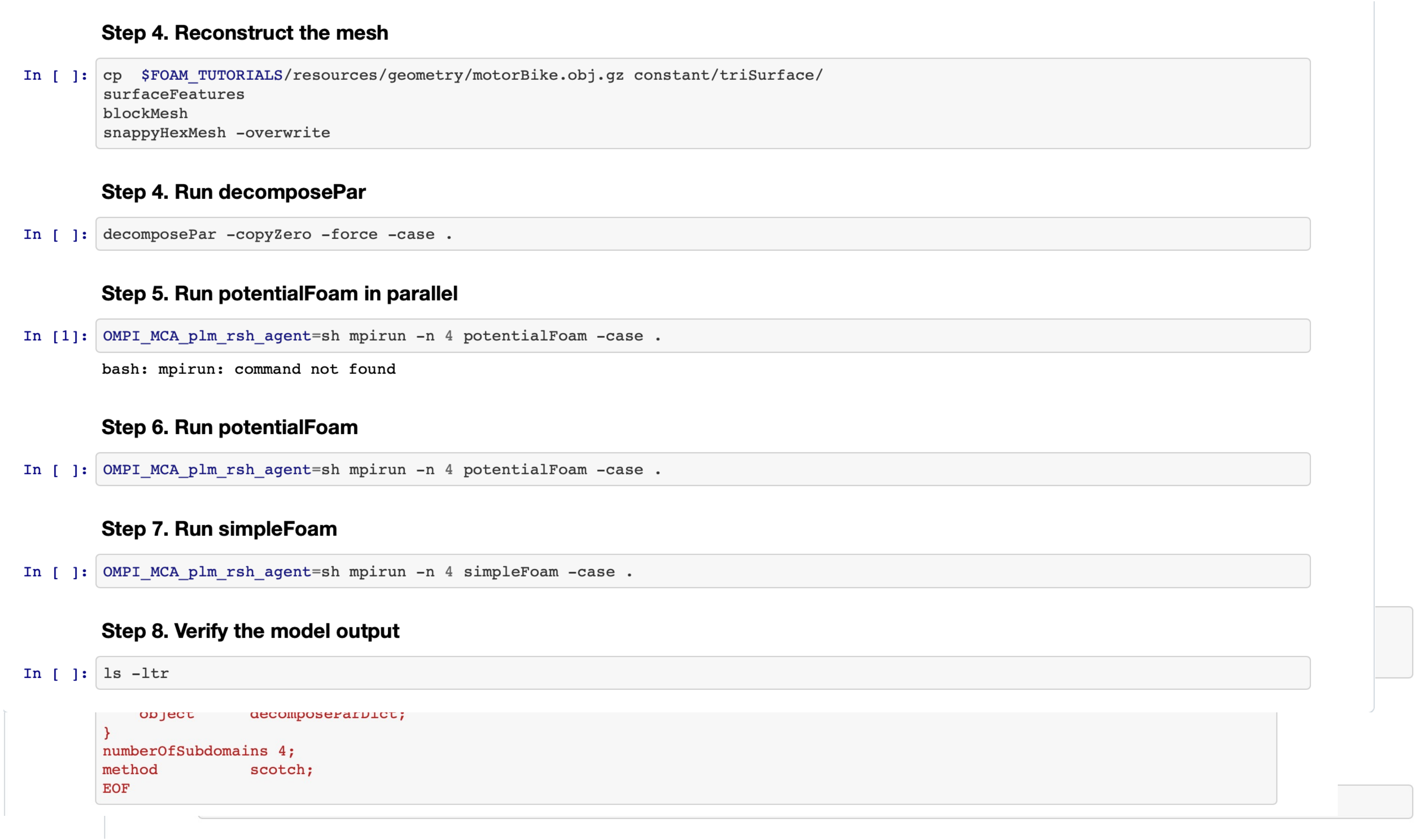
re that task was

Use Case 2 – notebook OpenFOAM

- Framework: ipython & CWL
- Application/workflow: OpenFOAM



Use Case 2 – Notebook OpenFOAM



Step 4. Reconstruct the mesh

In [ ]:
```
cp  $FOAM_TUTORIALS/resources/geometry/motorBike.obj.gz constant/triSurface/
surfaceFeatures
blockMesh
snappyHexMesh -overwrite
```

Step 4. Run decomposePar

In [ ]:
```
decomposePar -copyZero -force -case .
```

Step 5. Run potentialFoam in parallel

In [1]:
```
OMPI_MCA_plm_rsh_agent=sh mpirun -n 4 potentialFoam -case .
```

```
bash: mpirun: command not found
```

Step 6. Run potentialFoam

In [ ]:
```
OMPI_MCA_plm_rsh_agent=sh mpirun -n 4 potentialFoam -case .
```

Step 7. Run simpleFoam

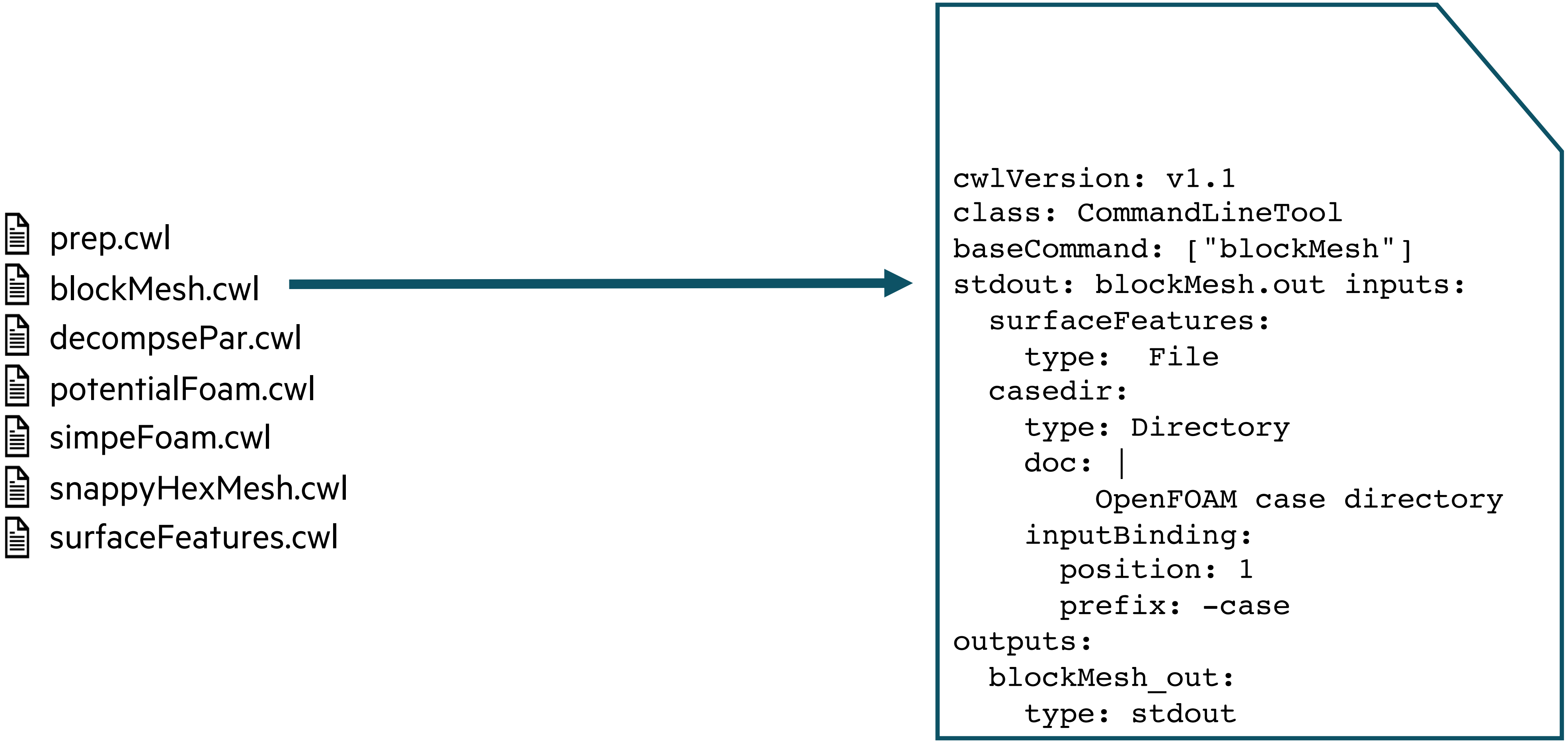
In [ ]:
```
OMPI_MCA_plm_rsh_agent=sh mpirun -n 4 simpleFoam -case .
```

Step 8. Verify the model output

In [ ]:
```
ls -ltr
```

```
    object      decomposeParDict;
}
numberOfSubdomains 4;
method          scotch;
EOF
```

Use Case 3 – CWL OpenFOAM



- prep.cwl
- blockMesh.cwl
- decompsePar.cwl
- potentialFoam.cwl
- simpeFoam.cwl
- snappyHexMesh.cwl
- surfaceFeatures.cwl

```
cwlVersion: v1.1
class: CommandLineTool
baseCommand: ["blockMesh"]
stdout: blockMesh.out inputs:
  surfaceFeatures:
    type:  File
  casedir:
    type: Directory
    doc: |
        OpenFOAM case directory
    inputBinding:
      position: 1
      prefix: -case
outputs:
  blockMesh_out:
    type: stdout
```

Use Case 3 – CWL OpenFOAM

Workflow declaration

```
#!/usr/bin/env cwl-runner
cwlVersion: v1.2
class: Workflow
# Workflow inputs.
inputs: casedir: Directory
outputs: []
```

Workflow steps

```
# Workflow steps.
# Typically, these are all in sequence, one after the other...
steps:
    …
    potentialFoam:
      run: of/potentialFoam.cwl
      in: casedir: casedir
         decomposePar:
           source: decomposePar/decomposePar_out
      out: [potentialFoam_out]
    simpleFoam:
      run: of/simpleFoam.cwl
      in:
        casedir: casedir
        potentialFoam:
          source: potentialFoam/potentialFoam_out
      out: [simpleFoam_out]
```

Wrap up

- A growing interest in HPC/AI workflows and marketplaces
- Modeling a workflow using a selected DSL requires attention
- Annotated workflows are easier to understand
- No clear leader – portals, community frameworks – no standards

Thank you

Jonathan "Bill" Sparks, jonathan.sparks@hpe.com
Ayad Jassim, ayad.jassim@hpe.com